General Disclaimer

One or more of the Following Statements may affect this Document

- This document has been reproduced from the best copy furnished by the organizational source. It is being released in the interest of making available as much information as possible.

- This document may contain data, which exceeds the sheet parameters. It was furnished in this condition by the organizational source and is the best copy available.

- This document may contain tone-on-tone or color graphs, charts and/or pictures, which have been reproduced in black and white.

- This document is paginated as submitted by the original source.

- Portions of this document are not fully legible due to the historical nature of some of the material. However, it is the best reproduction available from the original submission.

SMM-UVSP OZONE PROFILE INVERSION PROGRAMS

(NASA-CR-175216) SMM-UVSP OZONE PROFILE INVERSION PROGRAMS Final Report (Visidyne, Inc., Burlington, Mass.) 74 p HC A04/MF A01 CSCL 04A

N84-20042

Unclas
G3/46 12503

H.J.P. Smith
Visidyne, Inc.
5 Corporate Place
South Bedford Street
Burlington, MA 01803

September 1983
Final Report

Prepared for:
Goddard Space Flight Center
Greenbelt, Maryland, 20771

VISIDYNE
5 CORPORATE PLACE ■ SOUTH BEDFORD STREET ■ BURLINGTON, MASSACHUSETTS 01803
(617)273-2820

TECHNICAL REPORT STANDARD TITLE PAGE

1. Report No. VI-697	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle: SMM-UVSP OZONE PROFILE INVERSION PROGRAMS		5. Report Date: September 83 6. Performing Organization Code
7. Author(s): Henry J. P. Smith		8. Performing Organization Report No. VI-697
9. Performing Organization Name and Address: Visidyne, Inc. 5 Corporate Place South Bedford Street Burlington, MA 01803		10. Work Unit No. 11. Contract or Grant No. NAS5-26086 13. Type of Report and Period Covered: Final Report
12. Sponsoring Agency Name and Address: NASA Goddard Space Flight Center Greenbelt, MD		14. Sponsoring Agency Code
15. Supplementary Notes		

16. Abstract

This report forms the documentation and user manual for the software used to invert the UVSP aeronomy data taken by the SMM. The programs are described together with their interfaces and what inputs are required from the user.

17. Key Words (Selected by Author(s))	18. Distribution Statement
Solar Maximum Mission, Ozone, Nesosphere, Inversion Technique	

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price*
UNCLASSIFIED		73	

*For sale by the Clearinghouse for Federal Scientific and Technical Information, Springfield, Virginia 22151.

TABLE OF CONTENTS

SMM-UVSP OZONE PROFILE INVERSION PROGRAMS

1.0 INTRODUCTION

The documentation for the computer programs used to process the Ultra-Violet Spectrometer Polarimeter (UVSP) data taken by the Solar Maximum Mission satellite (SMM) to obtain the profiles of ozone in solar occultation is presented in this report. The next section presents an outline of the experiments performed together with a brief description of the technique used to invert the occultation data. Section 3 describes the main features of the programs and gives a pictorial view of how they link together. The final section presents a brief discussion of the results of this effort. Each of the individual programs and subprograms is described in Appendix 1 with the flow chart and a listing. Appendix 2 provides a user's manual with complete instructions for the use of the programs. Appendix 3 contains a description of the geometry of the occultation and how it is computed.

2.0 DESCRIPTION OF EXPERIMENT

The UltraViolet Spectrometer Photometer (UVSP) aboard the Solar Maximum Mission (SMM) satellite was used to obtain profiles of the atmospheric ozone in occultation. The instrument was programmed to use a given slit and to measure the magnitude of the solar flux that was transmitted through the atmosphere as the sun was appearing or disappearing behind the earth. A more complete description of the experiment is given in Reference 1.

In order to perform the inversion of the occultation data to obtain the ozone profile, two pieces of data must be used. The first is obviously the occultation profile itself; the second is the satellite ephemeris. The SMM was not capable of including a description of its location on the down-linked data stream sufficiently accurate for the purposes of inversion. Therefore, it was necessary to obtain the ephemeris from other sources namely from the satellite emphemeris group at Goddard Space Flight Center. The ephemeris must then be melded with the occultation profile with care to ensure that the two independent time streams match correctly. This in fact turned out to be the most difficult part of the effort.

The basic method used for the inversion of the occultation data starts from the relation

$$\tau = \exp(-2\sigma_\lambda \int [O_3]dl) \qquad (1)$$

where is the transmittance through the tangent path, is the ozone cross section at the wavelength of operation , and $[O_3]$ is the ozone concentration profile. The integral is to be taken over half the tangent path, that is from the tangent height to infinity or in practice, to some high altitude to be determined. This method of inversion has been discussed several times and we refer the interested reader to References 1 and 2. Equation 1 may be recast to a more tractable format

$$\int [O_3] dl = \ln(1/\tau)/2\sigma_\lambda \qquad (2)$$

This is a relatively simple linear integral equation and may be solved by standard matrix methods. It may also be pointed out that this method is entirely equivalent to a linear multiple regression for the concentration profile.

3.0 OVERALL DESCRIPTION OF THE INVERSION PROGRAMS

The inversion algorithms used for this effort were developed and run on a Digital Equipment Corporation (DEC) computer, a PDP-11/23 running the UNIX operating system. The programs were all written in Fortran 77 but were written such that a Fortran IV compiler with file commands (OPEN and CLOSE) could also be used. In fact the initial development was done on the DEC PDP-11/03 running RT-11 and using the DEC Fortran IV compiler. For those without a Fortran 77 compiler we describe in the appropriate places what few and minor changes must be made to run these programs.

The SMM UVSP occultation data used in this effort was provided on nine track tape in standard DEC format. The files were binary data with 512 bytes or 256 PDP-11 words per block. The first block of a file for a given experiment contained information about the experiment such as the date, time, wavelength, etc. The second block was a header block for the next logical record which consisted of sixteen file blocks, containing the digitized output of the counters measuring the solar flux as observed by the system. If there was more than one logical record for the experiment of interest, each such logical record of sixteen blocks of counter data was preceded by a header block as well. The header blocks contained such information as the start time of the record, end time, etc. A more complete description of the format of the UVSP final data tapes will be given in the description of the programs that read them.

The ephemeris data was also provided on nine track tape in DEC format. This data consisted of double precision floating point numbers for the most part with the remainder being integer descriptions of dates and times, etc. In order to use the ephemeris data in an optimum manner for the UVSP inversion it was necessary to decrease somewhat the time precision of the data. The data is given at one second intervals. The UVSP data is presented with a much finer time resolutions and it is necessary to provide some form of interpolation on the ephemeris in order to obtain the satellite position accurately at an arbitrary time. It was found that the emphemeris time intervals of one second was too fine for convenience since a third order spline fit to the ephemeris was selected as the most appropriate interpolation method.

In Figure 1, we give an overview of the set of programs used for the inversion. They may be easily be combined by using any of the usual command string interpreters on modern operating systems such as the UNIX Shell or RT-11's CSI. The following paragraphs describe the basic features of these programs and their interactions with each other. The detailed descriptions are left for Appendix 2.

The general flow of the analysis is shown in Figure 1. The two data streams are shown at the top of the Figure as files with extension *.AO for the ephemeris data and *.FD for the occultation data. (Note that the * is the usual DEC convention for a "wildcard" filename, i.e. it stands for any name.) These are processed by the data handling programs AOTOPT and DATPGM respectively in order to produce more convenient data files concentrated on the time frame of the occultation itself. These files are given the files extensions *.PT and *.OC respectively. The program TANSMM then reads both files to produce another data file which contains the occultation profiles as a function of the acutual tangent point location expressed as a vector with components height, latitude and longitude. This is written on a file with extension *.TH. The latter file contains more data than needed for the analysis of Equation 2. Thus the program BINS is used to compress and smooth the data into a more convenient format on the files *.BN. Finally, the program REAL03 performs the required inversion based on Equation 2.

The program AOTOPT was written to read the ephemeris tape and write a new file containing only those times around the occultation period and at ten second intervals. In order to be sure that the spline interpolation worked properly, its output at ten seconds was compared with the same for five second

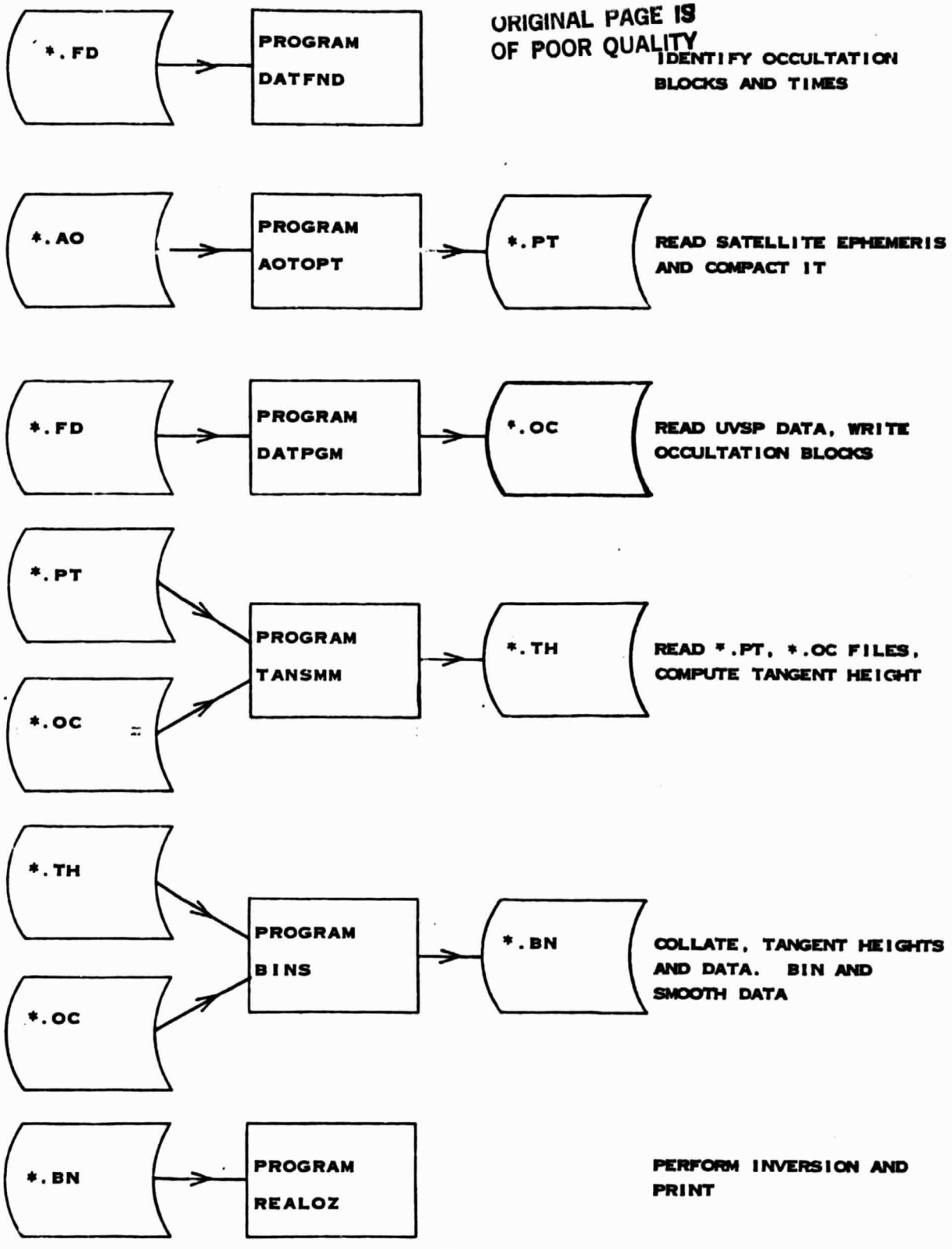


FIGURE 1. OVERVIEW OF INVERSION PROGRAMS

intervals for a number of selected cases. No appreciable difference was detected. Thus we concluded that the satellite ephemeris is well described by a spline fit at ten second intervals.

In practice, it is more convenient to read all the ephemeris files off the tapes and store them on disk. We have done so and give the files the DEC file extension *.AO where * is the usual DEC wildcard convention for the file name. Note that each of the UVSP experiments was given a sequential number with the file name starting with the letter "V", e.g. "V00513". Thus the ephemeris data for this experiment would have the file name "V00513.AO". The program "AOTOPT" creates the shorter ephemeris file "*.PT" following the convention just described.

The UVSP "final data" files (*.FD) were also read from tape to disk. The program DATPGM provides a listing of the experiment and block header in printed format. It also compresses the file to contain only the data in two blocks around the occultation on the file *.OC. The user is asked for the block location containing the start of the occultation. This must be known before the program is run. Another program (DATFND) based on the structure of DATPGM, was used to enquire for these quantities by looking through the *.FD files. It would be possible to automate this procedure but this was not found necessary for the present amount of data. The data for the occultation frames is then written on the files *.OC for use by the progrms TANSMM and BINS.

The program TANSMM reads the ephemeris files *.PT and performs a cubic spline fit to each of the three Cartesian coordinates of the orbit. The spline is then used to interpolate to get a precise position of the sensor at each of the times at which a count was recorded. The timing data is read by TANSMM for the files *.OC as written by DATPGM. The solar position is obtained by using a Chebychev polynomial fit to the solar ephemeris for the year 1980 as published by the U.S. Naval Observatory[3]. This figure of the earth is also approximated by an ellipsoid of revolution[4]. It was found that this simple form was entirely adequate for current purposes since the excursions of the measured figure of the earth from this figure were smaller than the precision of the experiment. The geometry of the line of sight was determined from the pitch, roll and yaw of the satellite and from the location of the center of the solar disk. A series of rotation matrices was used to transform from one coordinate system to another. A simple vector relation was then used to determine the tangent height location.

The program BINS was used to perform a seven point running average to smooth the data. In addition, the amount of data was cut to include only the region immediately around the occultation. For convenience in using the relatively small address space of the PDP-11, the data was further compressed by performing a binning of the data. The number of points to be averaged into each bin is a user input option. In practice it was found that three was a good number to use for this purpose. The output of these procedures was written on files *.BN.

The final program used in this procedure is called REAL03. It reads the binned data files *.BN and performs the inversion following the algorithm for the solution of Equation 2. The output is printed and no output disk file is made. For this purpose, we used the UNIX redirection features to good effect.

4.0 DISCUSSION

The programming effort described in this report shows the power of modern small computers which may be used for purposes for which only a large mainframe could suffice only a few years ago. Indeed most of the work described here could have been carried out on many of the latest generation of personal computers. The only problem would have been the use of the nine track tape drive for the data files. Even this could have been resolved easily by using a machine with a tape drive and down-loading the required data over a communications link.

The inversion technique employed allows a very simple algorithm to be used. A more complicated situation such as several species absorbing at the wavelength of interest would have been quiet a bit more complicated but still tractable through the use of least squares methods. The main limitation arises from the relatively small address space allowed by the PDP-11. It might be pointed out that some of the new 16-bit microprocessors have a much larger address space and this limitation would not apply.

REFERENCES

1. A.C. Aikin, B. Woodgate, and H.J.P. Smith, "Atmospheric Ozone Determination By Solar Occultation Using The UV Spectrometer on the Solar Maximum Mission", Applied Optics, Vol. 21, p. 2421 (July, 1982).

2. H.G. Linder, "Data Processing Plan For Solar Maximum Mission-A (SMM-A)", NASA X-565-78-20, (April, 1979).

3. U.S. Naval Observatory, "Aids For Computers", (1980).

4. W.A. Heiskanen and F.A. Vening Meinesz, "The Earth And Its Gravitational Field", McGraw-Hill (1958).

APPENDIX A

PROGRAM DESCRIPITONS

In this appendix we present the write-up and descriptions of each of the programs and subprograms used in the ozone inversion. An attempt has been made to follow the order of use in performing an inversion of a given set of occultation data.

1. PROGRAM DATFND

This program is used to search through the blocks of the given final data file Vxxxxx.FD., where Vxxxxx is the experiment number of the occultation of interest. (Each UVSP experiment had been assigned such a number as it was accepted for inclusion in the daily SMM experiment program). The user is prompted to enter this number namely "Vxxxxx"; for example it might be V01815. The number is then concatenated with the file extension ".FD" and the corresponding final data file is opened. The user is prompted again for the number of a block to be examined as to whether it contains the beginning of the occultation or not. The answer to this question is immediately apparent when the output is listed on the crt screen or printed on a hardcopy terminal. If it is the proper block the user is to make a note of it and to either end the execution of DATFND or to request another experiment file. If it is not an occultation block, one needs to input another candidate block number and continue the search until the occultation block is found. Note that a UVSP block consists of 256 sixteen bit integers and an occultation can extend over two such blocks but not more. Thus what one wants is the nubmer of the first of these two blocks.

As pointed out in the main body of the text it would be possible to program this procedure but it was not feit worthwhile for the current amount of SMM data. When the SMM is repaired in-orbit, the amount of data should undergo a dramatic increase. It will be necessary then to provide an automatic procedure for detecting the start of the occultation.

2. PROGRAM DATPGM

The program DATPGM is used to read the file header block and the

"page" header block to obtain the pointing and timing information for the given experiment. The occultation block number is then used to locate the two contiguous blocks which contain the actual occultatión profile.

The user is prompted for the file specification which is the experiment number Vxxxxx discussed in the previous program description for DATFND. The experiment number read is then concatenated with the final data extension ".FD." The next prompt is for the "header block number" to which one replies with an integer giving the block number of the "page" (ihblck). One is then prompted to give the number of the actual block (iblock) on which the occultation starts (found by using DATFND) and the nubmer of the actual page (ipage). It should be noted that there is one page header block for every sixteen blocks of UVSP count data. Thus the very first header block would have ihblck=2 and ipage=1. The second header block would have ihblck=18 and ipage=2 and so forth. The .FD file is then opened and the very first record (the file header record) is read. Various quantities are printed out for the user's reference and some floating point numbers are computed, for example the experiment start and stop times and the wavelength defining the spectrometer scan. The page header block is then read and more quantities are printed for reference and the pointing angles (pitch, roll and yaw) are computed together with start and stop times of the page. Note that these quantities have been read into two different arrays ismm and ismm4. The first array is for 2 byte integers and the second is for 4 byte inters.

The actual blocks containing the occultation are then read into arrays ismm(256) and ismm1(256) and, by an equivalence statement, the full array of the two blocks containing the occultation profile, ismmfl(512), is obtained simultaneously. The final data file is then closed.

The experiment number is concatenated with the string ".OC" and a file with this name is created as a sequential formatted file. Note that this file may be printed for the user's convenience if he so chooses. The floating point array data(512) is created by setting it equal to the array ismmfl(512) in a do loop. The maximum value of the array data(512) is then found by a call to FUNCTION RMXMN and the array data(512) is normalized using this maximum value. The values of the timing and angular information needed for the inversion are then written to the .OC file followed by the array data.

3. PROGRAM AOTOPT

This program reads the ephemeris tape and prepares the shortened form of the ephemeris for use in the spline fit. The File names are obtained in the same way as outlined for DATFND and DATPGM. The ephemeris has the extension ".PT" and the short ephemeris has the extension ".PT". The user is prompted for the start and stop times of interest in seconds of day. The .AO file is then read and for every tenth second the Cartesian coordinates are written to the .PT file together with the actual time in seconds. For convenience some output is printed as the procedure is followed. This may be kept as record of the process.

4. PROGRAM TANSMM

This is the most complicated of the programs discussed in this report. Its purpose is to read the two files *.PT and *.OC to obtain the profile of the data, not as a function of time but as a function of the tangent height of the ray path to the point on the sun that was being observed through the spectrometer slit in the given experiement. As mentioned in the main text, to do this it is necessary to have an accurately matched set of the two time series, namely the UVSP occultation data and the satellite ephemeris.

The user first supplies the familiar experiment number and this is concatenated with the .OC extension. The timing and anglular information is then read off this file and it is closed. The .PT file is then opened and the ephemeris data is read by calling subroutine EPHTST. This routine performs the calculation of the spline coefficients by calling the SPLINE subroutine. More details will be given in the appropriate section.

The pointing information is given with respect to the solar axis of rotation and thus this direction must be calculated and stored in the array solaxs(3). Seven of the .FD files we received had timing errors on them and an offset of 262.144 secs had to be subtracted. This is done in a do loop where a test is made to ascertain whether the present case is one of the seven.

The main loop of the program over each of the 512 data points contained in the occultation is then entered. The first action taken is to compute the Cartesian position of the sun center by a call to the solar ephemeris subroutine SOLEPH. The Cartesian satellite position at the current time is then computed by using the cubic spline evaluating function SEVAL. The radial posi-

tion of the satellite is then computed by a call to the subroutine LATLON, and the unit vector from the satellite to the solar center is obtained by a call to the subroutine UNIVEC. The rotation matrices needed are then calculated in subroutine ROLLIT, and the tangent height of the satellite-solar center path is obtained from a call to subroutine TANHGT. The first rotation is performed by a call to subroutine TANHGT. The first rotation is performed by a call to subroutine ROTMTX and this is followed by a translation. The next rotation precedes the calculation of the tangent point coordinates. The transformation process is then reversed by two calls to subroutine ROTMTX followed by a translation in the reverse sense to the preceding one and the final rotation is then processed. The calculation of the actual tangent point coordinates for the current sight path in subroutine TANHGT is preceded by the determination of the appropriate unit vector in subroutine UNIVEC. Finally, the figure of the earth is used to compute an accurate value for the height of the tangent point. To provide some kind of record of the computation the values for every tenth point obtained are printed.

When the loop has finished a new file is created with the extension ".TH" as a sequential, formatted file which may be printed if desired. The quantities printed include the latitude and longitude of the last tangent point computed and the array of tangent heights, htan(512). This file is then closed and the program ends.

5. PROGRAM BINS

Program BINS is used to smooth the data and to decrease the size of the data file used for the inversion. The full 512 points analyzed by the program TANSMM are far more than needed to describe the occultation adequately. Experience has shown that 150 points are sufficient for that purpose. Also the inversion program for convenience, always assumes that the occultation profile is presented in numerical order in the tangent height. For the raw data this ordering hold for the dawn cases but the order must be reversed for the dusk cases.

The user is prompted for the file name in the usual manner. The ".TH" File is opened and the latitude and longitude are read and printed. The user is then prompted for the type of occultation, dawn or dusk. The tangent height profile is then read in the appropriate order from the ".TH" file and

the file is closed. The ".OC" file is then opened to read the actual normalized data counts that correspond to the tangent height profile just read from the ".TH" file.

The program then prompts for the number of points per bin desired. For most cases we have found that three is the proper number to choose since this allows an in-memeory matrix inversion to be performed for the profile inversion. Since the occultation profile may occur anywhere within the 512 points, it is desirable to shorten the profile to simplify the data handling. The program prompts the user for the number of points to be skipped at the beginning of the 512 point block of data and for the number at the end of the block. The user may use the printout of the TANSMM program as an aid in deciding which points contain the actual profile. Care must be taken because the dusk profiles have been reversed as described above.

After closing the ".OC" file, a new output file is created with the extension ".BN". This contains the binned data points and the corresponding tangent heights averaged over the number of binning points (usually three). The file is closed before the program ends.

6. PROGRAM REALOZ

This program performs the actual inversion of the occultation profile to obtain the profile of the ozone as observed by the UVSP. Using the filename requested, the program opens the corresponding ".BN" file and reads its contents before closing it. It then prompts for the ozone absorption cross section for the wavelength of the experiment. The geometry matrix is initialized to zero and other geometric quantities are defined. The main loop to calculate the geometry matrix is then entered and the triangular matrix is computed in the linear array "DELS1(2601)" which corresponds to a matrix of any size up to 51 by 51. The matrix is computed as an equivalent linear array to allow the dimensions to be changed easily. The matrix inversion routine MINV is then called, the inverse is computed and returned in the same array DELS1. The required matrix multiplication is performed by calling the routine GMPRD and the ozone profile is printed before the program ends.

7. DOUBLE PRECISION FUNCTION DOT

This function computes the dot product of two Cartesian vectors and returns the value of the product. It is called by subroutine ROLLIT.

8. SUBROUTINE ELLIPS

This subroutine is called by the program TANSMM to compute the height, latitude and longitude of the calculated tangent point which is described as a Cartesian three-vector. The elliptic figure of the earth is used to assure the correct altitude.

9. SUBROUTINE EPHTST

This routine provides the spline fit of the satellite ephemeris. It is called by the program TANSMM. The first step is to initialize the various arrays to zero. The ephemeris data is then read from the file (.PT) which was opened in TANSMM. The subroutine SPLINE is then called for each of the three Cartesian coordinates of the satellite position vector. This provides a cubic spline fit which will be used in function SEVAL as called in TANSMM. Finally, the file is closed.

10. Subroutine GMPRD

This routine is adapted from the IBM Scientific Subroutine Package. It provides the generalized capability to multiply any two compatible matrices. It is called from the program REALOZ to multiply the inverted geometry matrix by the derived observation column vector to obtain the ozone profile. As the routine is heavily commented, its operation should be quite clear.

11. SUBROUTINE LATLON

This routine calculates the radius, latitude and longitude of any point given its Cartesian coordinates.

12. SUBROUTINE MINV

This IBM Scientific Subroutine Package routine provides the inverse

of any square matrix using the Gauss-Jordan method. The matrix is to be passed in the calling sequence as an equivalent linear array to enable changing dimensions easily. The comments in the routine should be sufficient for rapidly following the structure of the routine. It is callled by the program REALOZ.

13. FUNCTION RMXMN

The purpose of RMXMN is to compute either the maximum of an array of values or the minimum of the same array. The choice is made by the value of the third parameter in the calling sequence. The method is straight forward and uses a brute-force sequence. It should not be used for a very large sized array.

14. SUBROUTINES ROLLIT

This routine is called from the program TANSMM and provides the rotation matrices required for the transformation of the various coordinate systems. Three matrices are computed. The first, "R1", moves the z-axis to the radius through the satellite. The calling program then performs the translation such that the origin is then at the satellite itself. The second matrix, "R2", provides the z-axis along the sight-path of the center of the UVSP slit. The third matrix, "R3", is used to position the plane of viewing with respect to the solar axis of rotation which is used by the sensor as a convenient axis of reference.

15. SUBROUTINE ROTMTX

This subroutine performs the transformation of a given vector by a given rotation matrix. It takes advantage of the properties of orthogonality of rotation matrices to enable the inverse transformation using the transpose of the matrix as passed in the calling sequence. Which type of tranformation is desired, is determined by the value of a flag passed through the calling sequence.

16. FUNCTION SEVAL

This double precision function evaluates the cubic spline interpolation using the spline coefficients computed with subroutine SPLINE.

17. SUBROUTINE SPLINE

This subroutine computes the cubic spline coefficients required for the interpolation of the satellite ephemeris. Its operation should be clear from the comments.

18. SUBROUTINE SOLEPH

The solar ephemeris for the year 1980 is computed by evaluating several Chebyshev polynomial fits published by the U.S. Naval Observatory. Note that it would be necessary to change the coefficients if one were to analyze data from another year. This would be a simple matter to do. One only has to obtain the coefficients from the Naval Observatory publication corresponding to the year of the data.

19. SUBROUTINE TANHGT

The Cartesian coordinates of the tangent point for a given line-of-sight are computed in double precision by this subroutine. The unit vector of the line-of-sight is passed in the calling sequence as well as the distance to the tangent point ("TPARAM"). The tangent point coordinates ("XTAN") are returned together with the single precision magnitude of the radius to the tangent point ("RTAN").

20. SUBROUTINE UNIVEC

This routine computes the unit vector from the satellite to the sun and returns it in the array ("EUNIT").

PROGRAM DATFND

```
      program datfnd
c
c     this program reads the "final data" files *.fd
c     in order to find by hand the blocks in which the
c     occultation occurs. the number of the block should
c     be noted for use in program datpgm
c
c     the following 2 statements must be changed for Fortran IV
      chararter *30 fname
      character*6 name
      integer*2 ismm(256)
1     continue
c     user selects the file name to be examined
      print*,'enter file specification'
      read*, name
      print*, name
c     filename is concatenated with ".fd" extension
c     this is unix way of doing it change for rt-11
      fname=name//'.fd'
      print*,fname
      open(9,status='old',form='unformatted',access='direct',
     1 recl=512,file=fname)
c     user enters the block number he wants to look at
2     print*, 'block number'
      read*, iblock
c     the block is read and written to the screen of the crt
      read(9,rec=iblock) ismm
      print 100, ismm
100   format(10i8)
c     user selects another block or case or stops
15    print*, ' type 1 for another block, 2 another file, 0
exit'
c     read the selection
      read*,ians
      if(ians.eq.1) go to 2
      if(ians.eq.0) go to 99
c     user selected another file to examine
      if(ians.ne.2) go to 15
      close(9)
c     user selected another case
      go to 1
c     user selects to finish
99    close(9)
      stop
      end
```

SUBROUTINE SPLINE

```
      subroutine spline(n,x,y,b,c,d)
      integer n
      real*8 x(n),y(n),b(n),c(n),d(n)
ccc
ccc the coefficients b(i),c(i), and d(i), i=1,2..,n are computed
ccc for a cubic interpolating spline
ccc
ccc s(x)=y(i) + b(i)*(x-x(i)) + c(i)*(x-x(i))**2 +
d(i)*(x-x(i))**3
ccc
ccc for x(i) .le. x .le. x(i+1)
ccc
ccc input
ccc
ccc n = the number of data points or knots((n.ge.2)
ccc x = the abscissas of the knots in strictly increasing order
ccc y = the ordinates of the knots
ccc
ccc output..
ccc
ccc b,c,d = arrays of spline coefficients as defined above.
cc
ccc using p to denote differentiation
ccc
ccc y(i) = s(x(i))
ccc b(i) = sp(x(i))
ccc c(i) = spp(x(i))/2
ccc d(i) = sppp(x(i))/6 (derivative from the right)
ccc
ccc the accompanying function subprogram seval can be used
cc to evaluate the spline
ccc
ccc
      integer nm1,ib,i
      real*8 t
ccc
      nm1 = n-1
      if (n .lt. 2) return
      if (n .lt. 3) go to 50
ccc
ccc set up tridiagonal system
ccc
ccc b = diagonal, d = offdiagonal, c = right hand side
cc
      d(1) = x(2) - x(1)
      c(2) = (y(2) - y(1))/d(1)
      do 10 i=2,nm1
      d(i) = x(i+1)-x(i)
      b(i) = 2.0d00*(d(i-1) + d(i))
      c(i+1) = (y(i+1)-y(i))/d(i)
      c(i) = c(i+1)-c(i)
   10 continue
```

SUBROUTINE SPLINE

```
ccc
ccc end conditions.  third derivatives at x(i) and x(n)
cc obtained from divided differences
ccc
        b(1) = -d(1)
        b(n) = -d(n-1)
        c(1) = 0.0d00
        c(n) = 0.0d00
        if (n .eq.3) go to 15
        c(1) = c(3)/(x(4)-x(2))-c(2)/(x(3)-x(1))
        c(n) = c(n-1)/(x(n)-x(n-2)) - c(n-2)/(x(n-1)-x(n-3))
      c(1) = c(1)*d(1)**2/(x(4)-x(1))
        c(n) = -c(n)*d(n-1)**2/(x(n)-x(n-3))
ccc
ccc forward elimimation
ccc
   15   do 20 i=2,n
        t = d(i-1)/b(i-1)
        b(i) = b(i) - t*d(i-1)
        c(i) = c(i) - t*c(i-1)
   20   continue
ccc
ccc back substitution
ccc
        c(n) = c(n)/b(n)
        do 30 ib=1,nm1
        i = n-ib
        c(i) = (c(i) - d(i)*c(i+1))/b(i)
   30   continue
ccc
ccc c(i) is now the sigma(i) of the text
ccc
ccc compute polynomial coefficients
ccc
        b(n) = (y(n) - y(nm1))/d(nm1) + d(nm1)*(c(nm1) +
2.0d00*c(n))
        do 40 i=1,nm1
        b(i) = (y(i+1) - y(i))/d(i) - d(i)*(c(i+1) + 2.*c(i))
      d(i) = (c(i+1) - c(i))/d(i)
        c(i) = 3.0d00*c(i)
   40   continue
        c(n) = 3.0d00*c(n)
        d(n) = d(n-1)
        return
ccc
   50   b(1) = (y(2)-y(1))/(x(2)-x(1))
        c(1) = 0.0d00
        d(1) = 0.0d00
        b(2) = b(1)
        c(2) = 0.0d00
        d(2) =0.0d00
        return
        end
```

FUNCTION RMXMN

```
      function rmxmn (nd,data,ixn)
      dimension data(1)
c     returns max value (ixn=1) or min value (ixn=0) of array
      rmxmn=data(1)
      if(ixn.eq.0) go to 200
      do 100 j=2,nd
      if (data(j).gt.rmxmn) rmxmn=data(j)
100   continue
      return
200   do 210 j=2,nd
      if(data(j).lt.rmxmn)    rmxmn=data(j)
210   continue
      return
      end
```

SUBROUTINE UNIVEC

```
      subroutine univec(xsat,xsun,eunit)
c      returns double precision unit vector
c      along the vector (xsun-xsat)
      real*8 xsat(3),xsun(3),eunit(3),xmag
      xmag=0.0d00
      do 10 i=1,3
      xmag=xmag+(xsun(i)-xsat(i))**2
   10 continue
      xmag=dsqrt(xmag)
      do 20 i=1,3
      eunit(i)=(xsun(i)-xsat(i))/xmag
   20 continue
      return
      end
```

SUBROUTINE TANHGT

```
      subroutine tanhgt(xsat,eunit,tparam,xtan,rtan)
c      returns the location of the tangent point
      real*8 d,xsat(3),eunit(3),tparam,xtan(3)
      real*8 rtan
      d=((xsat(2)*eunit(3)-xsat(3)*eunit(2))**2+
     1   (xsat(3)*eunit(1)-xsat(1)*eunit(3))**2+
     2   (xsat(1)*eunit(2)-xsat(2)*eunit(1))**2)
      tparam=dsqrt(xsat(1)**2+xsat(2)**2+xsat(3)**2-d)
      d=dsqrt(d)
      do 10 i=1,3
      xtan(i)=xsat(i)+tparam*eunit(i)
   10 continue
      rtan=sngl(d)
      return
      end
```

SUBROUTINE CROSS

```
subroutine cross(x,y,xy)
real*8 x(3),y(3),xy(3)
xy(1) = x(2)*y(3) - x(3)*y(2)
xy(2) = x(3)*y(1) - x(1)*y(3)
xy(3) = x(1)*y(2) - x(2)*y(1)
return
end
```

PROGRAM TANSMM

```
      program tansmm
c
c     program to compute the tangent points for
c     the occultation blocks
c
      common/xyz/tt(50),xx(50),yy(50),zz(50),xc1(50),xc2(50),xc3(50),
     xyc1(50),yc2(50),yc3(50),zc1(50),zc2(50),zc3(50)
      real*8tt,xx,yy,zz,xc1,xc2,xc3,yc1,yc2,yc3,zc1,zc2,zc3
      integer*2imonth,iout,icheck,idayyr
c       the following statement is a UNIX space saver
c       it may be replaced by a simple dimension statement
c       for use on other systems
c
      automatic htan(512)
      real*8tstart,tblock,t,xsun(3),xsat(3),eunit(3),xtan(3),tparam,
     xtinc,srad,slat,slon,pitch,yaw,roll,xdum(3),ydum(3),r1(3, 3),
     x r2(3,3)
      real*8solaxs(3),aphi,bphi,tphi,eclptc,r3(3,3),soltlt,radg,sundec,
     xpi,sunzen,rtan,tanlat,tanlon
      character*30 fname
      character *6 name
      character*6 fxtim(7)
c       the following experiment numbers need special offsets
c       thus they are singled out in a data statement
      data fxtim/'V11134','V11140','V11144','V11171','V11172',
     1  'V11173','V11174'/
      data eclptc/23.433333333d00/
      data soltlt/7.25d00/
      data aphi/73.666666666d00/
      data bphi/1.3958333333d-02/
      radg = datan(1.d00)/45.d00
      pi = 180.d00*radg
23000 continue
c      user supplies the file or experiment name
      print*,'enter file specification'
       read *, name
       print*,name
c       following statement concatenates extension to filename
       fname=name//'.oc'
       print*,fname
c       open the file with the occultation data
       open(9, status='old',file=fname,access='sequential',
     1  recl=80,form='formatted')
c       file needs to be placed at beginning of information
       rewind 9
c       read the date and other time values
      read(9,10)idayyr,imonth
      read(9,20)tstart,tblock,pitch,yaw,roll,tinc
      close(9)
c       concatenate to get the partial ephemeris data
      fname = name//".pt"
```

```
      open(9,form = "formatted",status = "old",file = fname)
      rewind 9
c      get the spline fit to the ephemeris
      call ephtst(tblock,npt)
      soltlt = soltlt*radg
      eclptc = eclptc*radg
      tphi = dble(1980-1850)+(dble(idayyr)+tblock/24d00)/365.24d00
      tphi = (aphi+bphi*tphi)*radg
c      compute the solar rotation axis
      solaxs(1) = dsin(soltlt)*dcos(tphi)
      solaxs(2) = dsin(soltlt)*dsin(tphi)*dcos(eclptc)-dcos(soltlt)*
     xdsin(eclptc)
      solaxs(3) = dsin(soltlt)*dsin(tphi)*dsin(eclptc)+dcos(soltlt)*
     xdcos(eclptc)
      iout = 1
      icheck = 0
      t = tblock
c      compute the special offset if necessary
      do 200 ii=1,7
      if (name .eq. fxtim(ii)) t=t-262.144/3.6e03
 200  continue
      iday1=idayyr
c      loop over the two occultation blocks (512 points)
      do 23003 i = 1,512
      idayyr=iday1
c      get the solar ephemeris for the solar location
      call soleph(t,idayyr,xsun,sundec)
      sundec = sundec*radg
c      use the spline fit to get the satellite location
c      by calls to the function  seval
      xsat(1) = seval(npt,t,tt,xx,xc1,xc2,xc3)
      xsat(2) = seval(npt,t,tt,yy,yc1,yc2,yc3)
      xsat(3) = seval(npt,t,tt,zz,zc1,zc2,zc3)
c      call latlon to get radius, latitude & longitude
      call latlon(xsat,t,srad,slat,slon)
c      get the unit vector to the sun center
      call univec(xsat,xsun,eunit)
c      call the rotation matrix forming routine
      call rollit(xsat,srad,slat,slon,pitch,yaw,roll,eunit,
     x solaxs,r1,r2,r3,sunzen)
 101  format(3(2x,1pg14.7))
c      call the tangent height calculator routine
      call tanhgt(xsat,eunit,tparam,xtan,rtan)
c      perform the rotation transformation
      call rotmtx(xsat,r1,xdum,1)
c      do a translation
      xdum(3) = xdum(3)-srad
c      do another rotation
      call rotmtx(xdum,r2,ydum,1)
c      compute the tangent point coordinates
      xtan(3) = tparam
      xtan(2) = -tparam*dsin(yaw)/dcos(yaw)
```

```
      xtan(1) = -tparam*dsin(pitch)/dcos(pitch)
c      perform two more rotations
      call rotmtx(xtan,r3,ydum,-1)
      call rotmtx(ydum,r2,xdum,-1)
      xdum(1) = -xdum(1)
      xdum(2) = -xdum(2)
      xdum(3) = srad+xdum(3)
c      do another rotation
      call rotmtx(xdum,r1,xtan,-1)
c      compute the unit vector
      call univec(xsat,xtan,eunit)
c      get the tangent point
      call tanhgt(xsat,eunit,tparam,xtan,rtan)
c      compute position on elliptical earth
      call ellips(xtan,t,rtan,htan(i),tanlat,tanlon,sundec,sunzen,
     xpi)
c      print every tenth point
      if(.not.(mod((i+9),10).eq.0))goto 23005
      print*,"htan(i)=",i,htan(i),tanlat,tanlon
23005 continue
      t = t+tinc
      iout = iout+1
23003 continue
c      open the tangent height file   *.th
      fname = name//".th"
      open(9,status = "new",file = fname,access = "sequential",recl =
     x81,form = "formatted")
c      write the tangent point coordinate on *.th
      write(9,10)iout
      write(9,30)tanlat,tanlon
      write(9,40)htan
      close(9)
      stop
10    format(16i5)
20    format(4d20.12)
30    format(8f10 .3)
40    format(8f10 .5)
      end
```

PROGRAM BINS

```
      program bins
c
c       this program performs a binning and averaging
c       of the occultation being considered
c
      integer*2
iout,j,jj,nptbin,nskip,istart,iend,i,nendsk,ipick
      real*8 tstart,tblock,pitch,yaw,roll,tinc
      real*4 htan(512),data(512),hbin(51),dbin(51)
      character*30  fname
      character *6 name
ccc   specify and open input file *.th
      print*,'enter file specification'
      read*, name
      fname=name//'.th'

open(9,status='old',access='sequential',form='formatted',
     1  recl=81,file=fname)
      rewind 9
ccc   read input and reverse array if necessary
      read(9,10) iout
      read(9,30) tanlat,tanlon
      print*,'iout =',iout,tanlat,tanlon
c       user inputs whether dawn or dusk case
      print*,'  dawn or dusk, dawn=1,dusk=2'
      read*,ipick
      if(ipick.eq.1) read(9,40,end=201)(htan(i),i=1,iout)
      if(ipick.eq.2) read(9,40,end=201)(htan(i),i=iout,1,-1)
01    close(9)
c       read the file containing the occultation data named
c            *.oc

       fname=name//'.oc'
      print*,fname
      open(8,status='old',access='sequential',form='formatted',
     1  recl=80,file=fname)
      rewind 8
      read(8,10) idayyr,imonth
      read(8,20) tstart,tblock,pitch,yaw,roll,tinc
      if(ipick.eq.1) read(8,41,end=202) (data(j),j=1,iout)
     if(ipick.eq.2) read(8,41,end=202) (data(j),j=iout,1,-1)
02    close(8)
ccc   set up bins
c       user supplies the number of points per bin
  50  print*,'enter number of points per bin'
      read*,nptbin
c       user supplies the number of points to skip
c       at the beginning of the raw data file *.oc
      print*,'enter number of points to skip at beginning'
     read*,nskip
c       user supplies the number of points to skip at
```

PROGRAM BINS

```
c          the end of the raw data file
          print*,'enter number of points to skip at end'
          read*,nendsk
          iout=iout-nendsk
          istart=nskip+1
c          prepare the file which is to contain the binned
c          occultation data in a file  *.bn
          fname=name//'.bn'
          open(7,status='new',access='sequential',form='formatted',
      1   recl=80,file=fname)
          write(7,42) tanlat,tanlon
          iend=istart+nptbin-1
          jj=1
   55     hbin(jj)=0.0
          dbin(jj)=0.0
c          compute the averages of the tangent heights and data
          do 60 i=istart,iend
          hbin(jj)=hbin(jj)+htan(i)
          dbin(jj)=dbin(jj)+data(i)
     60   continue
          hbin(jj)=hbin(jj)/nptbin
          dbin(jj)=dbin(jj)/nptbin
          write(7,43) jj,hbin(jj),dbin(jj)
          istart=istart+nptbin
          iend=iend+nptbin
          if(iend.gt.iout) go to 99
          jj=jj+1
          go to 55
99        continue
          close(7)
          stop
10       format(16i5)
20         format(4d20.12)
30       format(8f10 .3)
40       format(8f10 .5)
41         format(8f10.7)
42         format(2f10.3)
43         format(i5,f10.3,f12.7)
          end
```

PROGRAM AOTOPT

```
      program aotopt
c
c       this program reads the SMM ephemeris data file
c       and pulls out the part of the ephemeris which
c       corresponds to the occultation period
c       this is placed on a file with extension '.pt'
c       the input comes from a file with extension '.ao'
c
      real*8 exptno,two,zdate(2),zsec(2),zday(2),zz(56)
      real*8 xdate,xday,xsec,xdut
      real*8 x(10),y(10),z(10),vx(10),vy(10),vz(10)
      character *30 fname,pname
      character *6 name
ccc   specify and open input file
1     continue
      print*,'enter file specification'
      read *,name
      if(name .eq. 'z') stop
c       the next statement concatenates name with the .ao
c       extension to get the ephemeris file required.
      fname=name//'.ao'
c       concatenate for the shortened data file *.pt
      pname=name//'.pt'
      print *,fname, pname
c       open the epehmeris file for reading
      open(9,status='old',access='direct',
     1 recl=512,file=fname)
c       get the times of interest from the input.
      print*,' input start and end time in sec'
      read*,tstart,tend
c       read *.ao to get experiment number etc.
      read(9,rec=1)exptno,two,
     1          (zdate(i),zday(i),zsec(i),i=1,2),zz
c       write it to standard output
      print 101,exptno,(zdate(i),zday(i),zsec(i),i=1,2)
      i=1
      ijk=1
c       open the shortened ephemeris file for output
      open(8,status='new',recl=252,file=pname,
     1 access='sequential',form='formatted')
c       start the read loop
2     continue
      i=i+1
c       read the ephemeris data
c       and print the data if time between tstart and tend
      read(9,rec=i,end=90)xdate,xday,xsec,xdut
     1          ,(x(k),y(k),z(k),vx(k),vy(k),vz(k),k=1,10)
      if(xsec.lt.tstart.or.xsec.gt.tend) go to 2
      print102,xdate,xday,xsec,xdut
c       write out every tenth data point
c       as epehemeris is needed every ten seconds
```

```
c          since the spline fit will be suffícient for
c          finer time detail
          k=1
          print 105,xsec,x(k),y(k),z(k)
          write(8,105) xsec,x(k),y(k),z(k)
105       format(f6.0,3(2x,d17.10))
          xsec=xsec+2.0
3         continue
          print 104, ijk
104       format(i10)
          ijk=ijk+1
          if(ijk .ge. 51) goto 10
          goto2
  90      print*, ' eof'
10        continue
          close(9)
          close(8)
          goto1
101       format(20x,'experiment number ',f9.1/,
     1    ' start  date = ',f9.1,2x,f5.1,'th day of year  at
',f9.1,
     2    'secs of day'/'   end  date = ',f9.1,2x,f5.1,'th day of
year  '
     3    'at  ',f9.1/)
102       format(/' date= ',f9.1,'  day= ',f5.1,'  secs= ',f9.1,
     1    '  dut= ',f14.5/10x,1hx,19x,1hy,19x,1hz/9x,2hvx,18x,2hvy,
     2    18x,2hvz)
103       format(i3,3f20.13/3x3f20.13)
          end
```

SUBROUTINE MINV

```
      subroutine  minv( a,n,d,l,m )
c.....................................................................

c
c         subroutine minv
c
c         purpose
c            invert a matrix
c
c         usage
c            call minv(a,n,d,l,m)
c
c         description of parameters
c            a - input matrix, destroyed in computation and
replaced by
c                 resultant inverse.
c            n - order of matrix a
c            d - resultant determinant
c            l - work vector of length n
c            m - work vector of length n
c
c         remarks
c            matrix a must be a general matrix
c
c         subroutines and function subprograms required
c           none
c
c         method
c            the standard gauss-jordan method is used. the
determinant
c            is also calculated. a determinant of zero indicates
that
c            the matrix is singular.
c
c
.....................................................................

c
c     subroutine minv(a,n,d,l,m)
      dimension a(1),l(1),m(1)
c
c
.....................................................................
c
c         if a double precision version of this routine is
desired, the
c         c in column 1 should be removed from the double
precision
c         statement which follows.
c
c     double precision a,d,biga,hold,dabs
```

SUBROUTINE MINV

```
c
c        the c must also be removed from double precision
statements
c        appearing in other routines used in conjunction with
this
c        routine.
c
c        the double precision version of this subroutine must
also
c        contain double precision fortran functions.  abs in
statement
c        10 must be changed to dabs.
c
c
.....................................................................
c
c        search for largest element
c
      d=1.0
      nk=-n
      do 80 k=1,n
      nk=nk+n
      l(k)=k
      m(k)=k
      kk=nk+k
      biga=a(kk)
      do 20 j=k,n
      iz=n*(j-1)
      do 20 i=k,n
      ij=iz+i
   10 if( abs(biga)- abs(a(ij))) 15,20,20
   15 biga=a(ij)
      l(k)=i
      m(k)=j
   20 continue
c
c        interchange rows
c
      j=l(k)
      if(j-k) 35,35,25
   25 ki=k-n
      do 30 i=1,n
      ki=ki+n
      hold=-a(ki)
      ji=ki-k+j
      a(ki)=a(ji)
   30 a(ji) =hold
c
c        interchange columns
c
   35 i=m(k)
      if(i-k) 45,45,38
```

SUBROUTINE MINV

```
   38 jp=n*(i-1)
      do 40 j=1,n
      jk=nk+j
      ji=jp+j
      hold=-a(jk)
      a(jk)=a(ji)
40       a(ji) =hold
c
c        divide column by minus pivot (value of pivot element is
c        contained in biga)
c
45    if(biga) 48,46,48
   46 d=0.0
        print*,'d=0',k,biga
      return
   48 do 55 i=1,n
      if(i-k) 50,55,50
   50 ik=nk+i
      a(ik)=a(ik)/(-biga)
   55 continue
c
c       reduce matrix
c
      do 65 i=1,n
      ik=nk+i
      hold=a(ik)
      ij=i-n
      do 65 j=i,n
      ij=ij+n
      if(i-k) 60,65,60
   60 if(j-k) 62,65,62
   62 kj=ij-i+k
      a(ij)=hold*a(kj)+a(ij)
   65 continue
c
c        divide row by pivot
c
      kj=k-n
      do 75 j=1,n
      kj=kj+n
      if(j-k) 70,75,70
   70 a(kj)=a(kj)/biga
   75 continue
c
c        product of pivots
c       print*,'minv',k,d,biga
c
      d=d*biga
c
c        replace pivot by reciprocal
c
      a(kk)=1.0/biga
```

```
   80 continue
c
c        final row and column interchange
c       print*,'got to r/c inter'
c
      k=n
  100 k=(k-1)
      if(k) 150,150,105
  105 i=l(k)
      if(i-k) 120,120,108
  108 jq=n*(k-1)
      jr=n*(i-1)
      do 110 j=1,n
      jk=jq+j
      hold=a(jk)
      ji=jr+j
      a(jk)=-a(ji)
  110 a(ji) =hold
  120 j=m(k)
      if(j-k) 100,100,125
  125 ki=k-n
      do 130 i=1,n
      ki=ki+n
      hold=a(ki)
      ji=ki-k+j
      a(ki)=-a(ji)
  130 a(ji) =hold
      go to 100
  150 return
      end
```

SUBROUTINE EPHTST

```
      subroutine ephtst(tblock,npt)
c
c       this routine reads the ephemeris data and
c       performs a spline fit for the cartesian coord-
c       ates which will later be evaluated for given times
c       by function seval
c
      common/xyz/t(50),xx(50),yy(50),zz(50),xc1(50),xc2(50),
     1  xc3(50),yc1(50),yc2(50),yc3(50),zc1(50),zc2(50),
     2  zc3(50)
      real*8 t,xx,yy,zz,xc1,xc2,xc3,yc1,yc2,yc3,zc1,zc2,zc3
     real*4 tt(50)
      real*8 tblock
c       do loop to zero the arrays as initial values
      do 5 i=1,50
      tt(i)=00.00
      xx(i)=00.d00
      yy(i)=xx(i)
      zz(i)=xx(i)
      t(i) =xx(i)
      xc1(i)=xx(i)
      xc2(i)=xx(i)
      xc3(i)=xx(i)
      yc1(i)=xx(i)
      yc2(i)=xx(i)
      yc3(i)=xx(i)
      zc1(i)=xx(i)
      zc2(i)=xx(i)
      zc3(i)=xx(i)
5     continue
      npt=1
c       read the ephemeris data to prepare for the spline fit
      do 100 k=1,50
      read(9,2000,end=300) tt(k),xx(k),yy(k),zz(k)
 2000 format(f6.0,2x,d17.10,2x,d17.10,2x,d17.10)
      npt=npt+1
c       convert units to current problem
      t(k)=dble(tt(k))/3600.d00
      xx(k)=xx(k)*1.d4
      yy(k)=yy(k)*1.d4
      zz(k)=zz(k)*1.d4
  100 continue
  300 npt=npt-1
c       call the spline routine three times
c       to fit each of the cartesian coordinates
      call spline(npt,t,xx,xc1,xc2,xc3)
      call spline(npt,t,yy,yc1,yc2,yc3)
      call spline(npt,t,zz,zc1,zc2,zc3)
c       print the values used
      print*,'npt=',npt
c
```

SUBROUTINE EPHTST

```
      do 800 kkk=1,npt
801   format(4g18.11)
      print 801,t(kkk),xx(kkk),yy(kkk),zz(kkk)
802   format(18x,3g18.11)
800   continue
      close(9)
      return
      end
```

PROGRAM REALOZ

```
      program realoz
c
c       this program inverts the geometry matrix
c       to obtain the ozone profile
c
      dimension ans(51),glamda(51),tau(51),lwork(51),mwork(51)
     1  ,htan(51),havg(51)
      dimension dels1(2601)
      character*30 fname
      character*6 name
c       arithmetic statement function for the figure
c        of the earth
      geoid(x)=6378.388*(1.-.3367003e-2*sin(x)**2+.7085e-5*
     1sin(x*2)**2)
      rdd=3.14159/180.
1     continue
      print*,'enter file specification'
      read*,name
c       concatenate for the binned data file
      fname=name//'.bn'
      open(9,status='old',access='sequential',form='formatted',
     1  recl=80,file=fname)
      read(9,42) tanlat,tanlon
42    format(2f10.3)
      do 57 jkq=1,51
      read(9,43,end=58) jjj,htan(jjj),tau(jjj)
43    format(i5,f10.3,f12.7)
57    continue
58    close(9)
      ntan=jjj
c     print*, tau
c       user supplies ozone cross section
      print*,'  input cross section... '
      read *,csx
      print *,'        cross section used = ',csx
c       zero out the matrix initially
      do 5 j=1,2601
5     dels1(j)=0.
c
      hupper=htan(ntan)
      dhtan=htan(ntan)-htan(ntan-1)
      ntan=ntan-1
      re=geoid(tanlat*rdd)
      rupper=re+hupper+dhtan
     ntan2=ntan*ntan
      i=0
      j=0
  8   continue
      j=i*ntan+i+1
      i=i+1
      rtan=htan(i)+re
```

PROGRAM REALOZ

```
      havg(i)=(htan(i)+htan(i+1))/2.
      h=htan(i)
      rtan2=rtan*rtan
      rupper=re+hupper+dhtan
      start=0.0
      k=i-1
      jj=j
9     k=k+1
      r=re+htan(k+1)
      sdist=sqrt(r*r-rtan2)
      dels1(jj)=(sdist-start)*1.0e-02
      start=sdist
      jj=ntan+jj
      if(jj. lt. ntan2) go to 9
      glamda(i)=0.0
      if(tau(i) .lt. 1.e-36) go to 15
      glamda(i)=0.5*alog(1./tau(i))/csx
15    continue
      if(j .lt. ntan2+1) go to 8
20    continue
21    continue
c     call the matrix inversion routine
      call minv (dels1,ntan,d,lwork,mwork)
c     call the matrix multiplication routine
      call gmprd (dels1,glamda,ans,ntan,ntan,1)
      print 991,tanlat,tanlon
991   format(2x,'tangent area latitude= ',f6.2,'   long.=
',f7.2)
      do2139 nn=1,ntan
2139  ans(nn)=abs(ans(nn))*1.0e-07
      do 2140 nn=1,ntan,3
      print992,
havg(nn),ans(nn),havg(nn+1),ans(nn+1),havg(nn+2),
     1          ans(nn+2)
992   format(3(0pf6.2,2x,1pe10.2,4x))
2140  continue
      stop
      end
```

```
      subroutine gmprd(a,b,r,n,m,l)
C.....................................................................
C
C     subroutine gmprd
C
C     purpose
C        multiply two general matrices to form a resultant
general
C        matrix
C
C     usage
C        call gmprd(a,b,r,n,m,l)
C
C     description of parameters
C        a - name of first input matrix
C        b - name of second input matrix
C        r - name of output matrix
C        n - number of rows in a
C        m - number of columns in a and rows in b
C        l - number of columns in b
C
C     remarks
C        all matrices must be stored as general matrices
        matrix r cannot be in the same location as matrix a
C        matrix r cannot be in the same location as matrix b
C        number of columns of matrix a must be equal to number
of row
C        of matrix b
C
C     subroutines and function subprograms required
CC         none
C
C     method
C        the m by l matrix b is premultiplied by the n by m
matrix a
C        and the result is stored in the n by l matrix r.
C
C
.....................................................................
C
C     subroutine gmprd(a,b,r,n,m,l)
      dimension a(1),b(1),r(1)
C
      ir=0
      ik=-m
      do 10 k=1,l
      ik=ik+m
      do 10 j=1,n
      ir=ir+1
      ji=j-n
```

SUBROUTINE GMPRD


```
      ib=ik
      r(ir)=0
      do 10 i=1,m
      ji=ji+n
      ib=ib+1
   10 r(ir)=r(ir)+a(ji)*b(ib)
      return
      end
```

PROGRAM DATPGM

```
      program datpgm
c
c       this program reads the occultation
c       data and culls out only the part of interest
c       which is contained on two DEC blocks
c
      real*4 strtm,stoptm,data(512)
      real*8 slamst,slamnd,xlamst,xlamnd,a0,a1,a2
      real*8 tstart,tblock,pitch,yaw,roll,pi,tinc,tblsec
     integer*2 ismm(256),ismm1(256),ismmfl(512)
      equivalence (ismm(1),ismmfl(1)),(ismm1(1),ismmfl(257))
     integer*2 i1,i2,i3,i4,i5,i6,i7,i13,i22,i23,i24,
    1  i25,i26,i27,i28,i29,i80,i81,i100,i109,i110,i111,
    2  i112,i113,ii4879(32),ii160(96),y
      integer*4 i2021,x,is67,is69
      integer*4 ismm4(128)
      real*4 i101,i103,i105,i107,i114,i116,i11C,i120,
    1  i122,i124,i126,i128,i130,i132,i134,i136,i138,i140,
    2  i142,i144,i146,i148,i150
      character*30 fname
      character *6 name
  1   continue
c       user supplies the filename without extension
      print*,'enter file specification'
      read *, name
      print*,name
c       open files required for this run
c        first the final data file
      fname=name//'.fd'
      print*,fname
ccc     read block number and file number
c       user supplies the header block number (2,18, etc)
      print*,'header block number'
      read*,ihblck
      print*,'iblock,ipage'
      read*,iblock,ipage
      open(9,status='old',form='unformatted',access='direct',
    1 recl=512,file=fname)
c       read the final data file
      read(9,rec=1)ismm
      read(9,rec=1) ismm4
c       print the data read
      print 100,ismm(10),ismm(11),ismm(12),ismm(13)
  100  format('  start time  ',i3,' ',i2,':',i2,':',i3)
     print 101, ismm(14),ismm(15),ismm(16),ismm(17)
  101  format('  stop time  ',i3,' ',i2,':',i2,':',i3)
     strtm=3600.*(ismm(11))+60.*(ismm(12))+ismm(13)
      stoptm=3600.*(ismm(15))+60.*(ismm(16))+ismm(17)
     print*, ' start time-secs ',          ismm(10),'  ',strtm
      print*, 'stop time -sec  ',ismm(14),'  ',stoptm
     print*,'  experiment type  ',ismm(60)
```

```
      print*,'  no. of detectors used  ',ismm(61)
      print*,ismm4(34),ismm4(35)
      print*, (ismm(izq),izq=65,72)
c      compute the actual wavelengths in angstroms
      xlamst=dble(ismm4(34))
      xlamnd=dble(ismm4(35))
      print*,'  xlamst =',xlamst,'      xlamnd =',xlamnd
     a0=1.9707373047d03
      a1=5.128015298d-3
      a2=2.8786d-10
      slamst=2.0*(a0-(xlamst+206.)*a1-((xlamst+206)**2)*a2)
     slamnd=2.0*(a0-(xlamnd+206.)*a1-((xlamnd+206.)**2)*a2)
     print*,'slamst  =',slamst,'slamnd  =',slamnd
      print*,'  no. of wld steps  ',ismm(78)
      print*,'  wld step size (dw)  '.ismm(79)
      read(9,rec=ihblck)
i1,i2,i3,i4,i5,i6,i7,x,x,y,i13,x,x,x,i2021,
     1  i22,i23,i24,i25,i26,i27,i28,i29,x,x,x,x,x,x,x,x,x,
    2  ii4879,i80,i81,x,x,x,x,x,x,x,x,x,i100,i101,i103,i105,i107,
     3  i109,i110,i11,i112,i113,i114,i116,i118,i120,
     4  i122,i124,i126,i128,i130,i132,i134,ii136,i138,
    5  i1440,i142,i144,i146,i148,i150,x,x,x,x,ii160
      idayyr=i29
      ihr=i25
      imin=i26
      isec=i27
      imsec=i28
c      compute roll pitch yaw etc from the .fd file
      pi = 4.d00*datan(1.d00)
      pitch = dble(ismm(23))/10.d00/3600.d00
      yaw = dble(ismm(24))/10.d00/3600.d00
      roll = dble(ismm(25))/100.d00
      pitch = pitch*pi/180.d00
      yaw = yaw*pi/180.d00
      roll = roll*pi/180.d00
c      compute start and end times
      tstart=dble(ihr)+(dble(imin)*6d1+dble(isec))/3600d0+
    1    dble(imsec)/3.6d6
c      get proper offsets
      if(ihblck.eq.2) iinc=iblock-3
      if(ihblck.eq.19) iinc=iblock-20
      if(ihblck.eq.38) iinc=iblock-38
      if(ihblck.eq.53) iinc=iblock-56
      if(ihblck.ne.2.and.ihblck.ne.19) print*,'error in header
block numb
     1  er'
c      compute proper block start time
      tblock=tstart+dble(.016*i101*256*(iinc))/3.6d3
      tinc=.016d00*dble(i101)/3600.d00
c      print the data for convenience
      print 200,i100
  200   format(1x,'no. of actual data points found  ',i5)
```

```
      print 201,i101
  201  format(1x,'mean time between data points in inner'
     1 'loop in units of 16ms', 1pe12.5)
       print 202,i103
  202  format(1x,'max. time gap found  ',1pe12.5)
       print 203, i105
  203  format(1x,'min time gap found',1pe12.5)
       print 204, i105
  204  format(1x,'standard deviation(sample)',1pe12.5)
      print 205,i109
  205  format(1x,'no. of inner loops',i5)
       print 206,i110
  206  format(1x,'count of next loop',i5)
       print 206,i111
       print 206,i112
       print 207,i113
  207  format(1x,'count of executions in this record',i5)
      print 208,i114
  208  format(1x,'mean time between inner loops', 1pe12.5)
      print 209, i116
  209  format(1x,'mean time between 2nd loops',1pe12.5)
      print 210,i118
  210  format(1x,'mean time between 3rd loops',1pe12.5)
      print 211,i120
  211  format(1x,'mean time between outer loops',1pe12.5)
      print 212,i122
  212  format(1x,'mean time between executions',1pe12.5)
      print 213,i124
  213  format(1x,'max time between inner loops',1pe12.5)
      print 214,i126
  214  format(1x,'max for next',1pe12.5)
       print 214,i128
       print 223,i130
  223  format(1x,'max for outer',1pe12.5)
       print 215,i132
  215  format(1x,'max for executions',1pe12.5)
       print 216,i134
  216  format(1x,'min time between inner loops',1pe12.5)
      print 217,i136
  217  format(1x,'min for next',1pe12.5)
       print 217,i138
       print 218,i140
  218  format(1x,'min for outer',1pe12.5)
       print 219,i142
  219  format(1x,'min for executions',1pe12.5)
       print 220,i144
  220  format(1x,'standard deviation for inner loop mean
time',1pe12.5)
       print 221,i146
  221  format(1x,'standard deviation for next loop',1pe12.5)
      print 221,i148
       print 222,i150
```

PROGRAM DATPGM

```
  222   format(1x,'standard deviation for outer loop',1pe12.5)
        print*,' i25,6,7,8,9  ',i25,i26,i27,i28,i29
c        now read the first block of the occultation
        read(9,rec=iblock) ismm
c        then read the second block
        read(9,rec=iblock+1) ismm1
        close(9)
c        prepare to open the *.oc
c        file for writing the occultation blocks
        fname=name//'.oc'
c        convert to floating point
        do 10 i=1,512
10      data(i)=ismmfl(i)
c       get max value of data array
        datmax = rmxmn(512,data,1)
c       normalize data array to datmax
        do 20 i=1,512
20      data(i) = data(i)/datmax
        open(9, status='new',file=fname,access='sequential',
     1  recl=80,form='formatted')
c        write the occultation data to the *.oc file
        write(9,251) idayyr,imonth
  251   format(2i5)
        write(9,252) tstart,tblock,pitch,yaw,roll,tinc
  252   format(4d20.12)
        write(9,253) data
  253   format(8f10.7)
        close(9)
        stop
        end
```

SUBROUTINE ROLLIT

```
      subroutine rollit(xsat,rsat,slat,slon,pitch,yaw,roll,
     1  eunit,solaxs,r1,r2,r3,sunzen)
c
c     this routine prepares the rotation matrices
c      for the coordinate transformations required
c
      real*8 xsat(3),rsat,slat,slon,pitch,yaw,roll,cos1,cos2,
     1  sin1,sin2,sunzen,phi4,phisun,solaxs(3)
      real*8 r1(3,3),r2(3,3),eunit(3),xunit(3),r3(3,3),yunit(3)
c      use the following convention for the sines and cosines
c     one = lat  :  two = long
      cos1 = dcos(slat)
      sin1 = dsin(slat)
      cos2 = dcos(slon)
      sin2 = dsin(slon)
c
c     set up lat-lon rotation matrix r1
c
      r1(1,1) = sin1*cos2
      r1(1,2) = sin1*sin2
      r1(1,3) = -cos1
      r1(2,1) = -sin2
      r1(2,2) = cos2
      r1(2,3) = 0.0d00
      r1(3,1) = +cos1*cos2
      r1(3,2) = +cos1*sin2
      r1(3,3) = sin1
c     use now the following convention
c     one = solar zenith angle   :   two = azimuth
c
c     transform unit vector
      call rotmtx(eunit,r1,xunit,1)
c     invert x and y axes
      xunit(1)=-xunit(1)
      xunit(2)=-xunit(2)
      phi4=datan2(xunit(2),xunit(1))
c     compute azimuth of solar rotation matrix
      cos2=dcos(phi4)
      sin2=dsin(phi4)
      cos1 = dot(eunit,xsat)/rsat
      sin1 = dsqrt(1.d00-cos1**2)
      sunzen=datan2(sin1,cos1)
c
c     set up zenith angle - azimuth rotation matrix r2
c
      r2(1,1) = +cos2*cos1
      r2(1,2) = sin2*cos1
      r2(1,3) = -sin1
      r2(2,1) = -sin2
      r2(2,2) = +cos2
      r2(2,3) = 0.0d00
```

```
      r2(3,1) = sin1*cos2
      r2(3,2) = sin1*sin2
      r2(3,3) = cos1
c     compute solar axis in satelite system
      call rotmtx(solaxs,r1,xunit,+1)
      xunit(1)=-xunit(1)
      xunit(2)=-xunit(2)
      call rotmtx(xunit,r2,yunit,+1)
      phisun=datan2(yunit(2),yunit(1))
      cos1=dcos(phisun)
      sin1=dsin(phisun)
c
c      now compute the actual rotation matrix r3
      r3(1,1)=cos1
      r3(1,2)=sin1
      r3(1,3)=0.0d00
      r3(2,1)=-sin1
      r3(2,2)=cos1
      r3(2,3)=0.0d00
      r3(3,1)=0.0d00
      r3(3,2)=0.0d00
      r3(3,3)=1.0d00
      return
      end
```

SUBROUTINE ROTMTX

```
      subroutine rotmtx(xin,rot,xout,iflag)
c
c     this routine transforms column vector xin to xout,
        under the rotation matrix  rot.....
c     if iflag less than zero the transpose of rot is used
c
      real*8  xin(3),rot(3,3),xout(3)
c
      do 5 i=1,3
         xout(i) = 0.0d00
5     continue
      if( iflag .lt. 0 ) goto 15
      do 10 i=1,3
         do 10 j=1,3
            xout(i) = xout(i) + rot(i,j)*xin(j)
10    continue
      return
15    continue
      do 20 i=1,3
         do 20 j=1,3
            xout(i) = xout(i) + rot(j,i)*xin(j)
20    continue
      return
      end
```

SUBROUTINE SOLEPH

```
      subroutine soleph(time,dayno,xsun,decl)
c
c       this routine computes the solar ephemeris
c       for the year 1980 using the U.S. Naval Observatory
c       chebyshev fit published in the Aids for Computors.
c       output is in earth-centered cartesian system
c
      real*8 time,xsun(3),crtasc(25),cdec(25),cdist(25)
     .,brtasc(25),bdec(25),bdist(25),x,pi,si,rtasc,decl,distau
      real*8 solrad
      integer*2 dayno
c       computed solar geocentric x,y,z
c       for dayno day of year 1980  -- input
c       for time in universal time  -- input
c       gives xsun in km
      data crtasc/
     1  61.3798984d00,11.8882784d00,0.0274567d00,0.0728274d00,
     2  0.0399380d00,0.1047579d00,-0.0307469d00,-0.0441609d00,
     3  0.0071710d00,0.0074422d00,-0.0020151d00,-0.0017746d00,
     4  0.0010209d00,0.0006818d00,-0.0003326d00,-0.0001905d00,
     5  0.0001304d00,0.0000686d00,-0.0000403d00,0.0000069d00,
     6  -0.0000085d00,0.0000025d00,0.0d00,0.0d00,0.0d00/
      data cdec/
     1  -13.485026d00,-2.371409d00,-22.511744d00,2.734964d00,
     2  6.654348d00,-0.370476d00,-0.499175d00,0.064997d00,
     3  0.073062d00,-0.038612d00,-0.036087d00,0.012058d00,
     4  0.008821d00,-0.003071d00,-0.001719d00,0.001055d00,
     5  0.000510d00,-0.000433d00,-0.000207d00,0.000130d00,
     6  -0.000014d00,0.000032d00,0.0d00,0.0d00,0.0d00/
      data cdist/
     1
1.98998373d00,0.00040082d00,-0.01629251d00,-0.00045237d00,
     2  0.00500746d00,0.00006182d00,-0.00042170d00,0.00000621d00,
     3  0.00000367d00,0.00000513d00,0.00000209d00,0.00000568d00,
     4  0.00000356d00,0.00000282d00,0.00000528d00,-0.00000141d00,
     5
0.00000392d00,-0.00000569d00,0.00000007d00,-0.00000566d00,
     6  -0.00000470d00,0.0d00,0.0d00,0.0d00,0.0d00/
c
c

x=(dble(dayno)+time/24.d00+.591d-03)/183.d00-1.005546448d00
      x=2.d0*x
      print*,'soleph x= ',x
      do 200 i=25,1,-1
      print*,'   i= ',i
      if(i.lt.23) go to 100
      brtasc(i)=0.0d00
      bdec(i)=0.0d00
      bdist(i)=0.0d00
```

SUBROUTINE SOLEPH

```
      go to 200
100   continue
      brtasc(i)=x*brtasc(i+1)-brtasc(i+2)+crtasc(i)
      bdec(i) = x*bdec(i+1)-bdec(i+2)+cdec(i)
      bdist(i) = x*bdist(i+1)-bdist(i+2)+cdist(i)
200   continue
      print*, '     brtasc',brtasc
      print*,'       bdec', bdec
      print*,'      bdist', bdist
      rtasc = 7.5d00*(brtasc(1)-brtasc(3))
      if(rtasc.gt.360.d00) rtasc = rtasc-360.d00
      decl = .5d00*(bdec(1)-bdec(3))
      distau = .5d00*(bdist(1)-bdist(3))
      solrad = 1.495985d08*distau
      pi = 4.d00*datan(1.d00)
      s1= solrad*(dcos(pi*decl/180.d00))
      xsun(1) = s1*dcos(pi*rtasc/180.d00)
      xsun(2) = s1*dsin(pi*rtasc/180.d00)
      xsun(3) = solrad*dsin(pi*decl/180.d00)
      print*,'  in soleph xsun= ',xsun
      return
      end
```

FUNCTION SEVAL

```
      real *8 function seval(n,u,x,y,b,c,d)
      integer n
      real*8 u,x(n),y(n),b(n),c(n),d(n)
ccc
ccc this subroutine evaluates the cubic spline function
cc
ccc  seval = y(i) + b(i)*(u-x(i)) + c(i)*(u-x(i))**2  +
d(i)*(u-x(i))**3
ccc
ccc where x(i) .lt. u .lt.x(i+1), using horner's rule
ccc
ccc if u .lt. x(1) then i = is used
ccc if u .gt. x(n) then i=n is used
ccc
ccc input.
ccc
ccc n = the number of data points
ccc u = the abcissa at which the spline is to be evaluated
cc x,y = the arrays of data abcissas and ordinates
ccc b,c,d = arrays of spline coefficients computed b. spline
cc
ccc if u is not in the same interval as the previous call, then a
ccc  binary search is performed to determine the proper interval.
ccc
      integer i,j,k
      real*8 dx
      data i/1/
      if (i .ge. n) i=1
      if (u .lt. x(i)) go to 10
      if (u .le. x(i+1)) go to 30
ccc
ccc binary search
ccc
   10 i = 1
      j = n+1
   20 k = (i+j)/2
      if (u .lt. x(k)) j = k
      if (u .ge. x(k)) i=k
      if (j .gt. i+1) go to 20
ccc
ccc evaluate spline
ccc
   30 dx = u - x(i)
      seval = y(i) + dx*(b(i) + dx*(c(i) + dx*d(i)
     return
      end
```

SUBROUTINE LATLON

```
      subroutine latlon(xsat,t,rsat,slat,slon)
c
c     routine computes radius, latitude and longitude
c     given cartesian coordinates as input
c
      real*8 xsat(3),slat,slon,rsat,rxy,t
      rsat = xsat(1)**2 + xsat(2)**2
      rxy = dsqrt(rsat)
      rsat = dsqrt(rsat+xsat(3)**2)
      slat = datan2(xsat(3),rxy)
      slon = datan2(xsat(2),xsat(1))
      return
      end
```

SUBROUTINE LATLON

```
      subroutine latlon(xsat,t,rsat,slat,slon)
c
c     routine computes radius, latitude and longitude
c     given cartesian coordinates as input
c
      real*8 xsat(3),slat,slon,rsat,rxy,t
      rsat = xsat(1)**2 + xsat(2)**2
      rxy = dsqrt(rsat)
      rsat = dsqrt(rsat+xsat(3)**2)
      slat = datan2(xsat(3),rxy)
      slon = datan2(xsat(2),xsat(1))
      return
      end
```

SUBROUTINE ELLIPS

```
      subroutine ellips(xtan,t,rtan,htan,tanlat,tanlon,
     1  sundec,sunzen,pi)
c
c       computes elliptical earth position
c
      implicit real*8 (a-h,o-z)
      real*4 htan
      call latlon(xtan,t,rtan,tanlat,tanlon)
      coshr=((dcos(sunzen))-dsin(tanlat)*(dsin(sundec)))/
     1 (dcos(tanlat)*(dcos(sundec)))
      sinhr=sqrt(1.-coshr**2)
      hr=datan2(sinhr,coshr)
      tanlon=((t/24.d00)*2*pi-pi)+hr
      re=6378.388*(1.-.3367003e-2*dsin(tanlat)**2+.7085e-5*
     1 dsin(tanlat*2)**2)
      htan=rtan-re
      tanlat=tanlat*57.29578
      tanlon=tanlon*57.29578
      return
      end
```

FUNCTION DOT

```
      double precision function dot(x,y)
c      computes dot product of vectors  x and  y
c                 returns as   dot
c
      real*8 x(3),y(3)
      dot = x(1)*y(1) + x(2)*y(2) + x(3)*y(3)
      return
      end
```

APPENDIX B

This appendix forms a brief user's manual for the programs described in the remainder of the report. It is intended to provide a user with all information necessary to be able to run the programs and obtain an inversion of the UVSP data for a single absorbing species such as ozone.

The first step is to get the SMM data into the computer in a usable format. The data was provided to Visidyne on magnetic tape. Our procedure was to read the files on this tape, copy them onto disk where they are much simpler to handle. If these programs are to be run on the UVSP data computers then this is already accomplished, otherwise the user must get the data into the main disk space or directory of the target machine. Each UVSP experiment has been assigned a number in the form "Vxxxxx" where V identifies it as a UVSP experiment and xxxxx represents the numerical order of the particular experiment. (For example, the experiment number V00512 was the very first one done for the purpose of ozone inversion.) It is convenient to name each file with its corresponding number. The UVSP "final data" files are therefore named with the usual DEC extension as "Vxxxxx.FD" with "FD" standing for "final data".

The second step is to obtain the accurate SMM ephemeris data from the satellite ephemeris group at Goddard Space Flight Center. This too will be provided on tape and the data must also be read and copies onto disk. The ephemeris data files are named "Vxxxxx.AO", following the convention described above, Note that it is necessary to be sure that the data is in a format appropriate for the target computer. It may be provided in one of several manufacturer's formats and it is best to avoid having to program conversions from one floating point binary format to another.

It is useful to point out once again that the two different time series involved (the final data "Vxxxxx.FD" and the ephemeris "Vxxxxx.AO") must be accurately matched in order to obtain a valid inversion. The actual occultation occurs over a period of only a few seconds so it is clear that this can be critical. In fact several occultations could not be analyzed because the two series could not be matched to give reasonable ozone profiles at all.

Having obtained the data as described above, the user must first run the program DATFND to identify which blocks of the *.FD files contain the

occultation. While it would certainly be possible to program the selection of these blocks, we chose not do so because of the relative number of experiments to be analyzed. It is necessary to note which blocks do contain the data for each occultation experiment to be analyzed and record them for later use.

The program DATPGM will prompt the user for the block numbers recorded from running DATFND. After this DATPGM will produce the shortened data file "Vxxxx.OC" and will print a set of parameters characterizing the experiment including such values as the wavelength and the start and end times of the record containing the occultation blocks. It is useful to keep this printout in order to know what values of the times to feed into the ephemeris program AOTOPT which will prompt for them. After reading these values, AOTOPT reads the long ephemeris file "Vxxxxx.AO" and writes to the shortened ephemeris file "Vxxxxx.PT", fifty satellite positions corresponding to a period of 500 seconds of time containing the occultation blocks within it. This was done for convenience in the use of the spline fitting and evaluation routines but a number other than fifty could just as well have been chosen as long as there were sufficient accuracy in the resulting spline fits.

The program TANSMM uses both the "Vxxxxx.OC" and the "Vxxxxx.PT" files to produce a file of tangent heights corresponding to the instantaneous lines-of-sight for each of the data points in the two occultation blocks (512). The user is prompted for the experiment name and for the cross section for the absorbing species at the wavelength of the instrument for this experiment (as printed out by program DATPGM). This data is written out to a file called "Vxxxxx.TH". This latter file is used by the binning and smoothing program "BINS" along with the file "Vxxxxx.OC" to produce another file of a size convenient for the inversion program (REALOZ) to analyze. This file is called "Vxxxxx.BN".

The reader will probably have noticed that the procedure described above is interactive and requires frequent responses from a user. This method of proceding was found more appropriate than a pure batch type of operation. Also, the process was broken up into several relatively small programs to simplify the use of a small computer. Most small systems do however provide for the definition of strings of commands which will eliminate the necessity of keying in these commands to run the individual

progams. We give two examples of such command strings, the first for the UNIX system and the second for the RT-11 operating system.

UNIX SHELL COMMAND

```
datpgm
aotopt
tansmm
bins
realoz
```

RT-11 COMMAND STRING INTERPRETER

```
R DATPGM
R AOTOPT
R TANSMM
R BINS
R REALOZ
```

The UNIX commands which we shall refer to as a "shell script" are to be written to a file using one of the editors and the file is to be declared as executable after it has been created. The shell script has been written with the assumption that the data files reside on the same directory as the programs. This could be easily rearranged for another directory structure by anyone familiar with the UNIX shell. Note that the shell script does not provide for the removal of the intermediate files produced by the procedure. If there is a shortage of disk space, one might want to add a command to remove these files from the disk before finishing. Also one might want to provide for the redirection of the REALOZ output either to a printer or to a file for later printing. As the programs stand, the output goes to the terminal.

The RT-11 CSI (Command String Interpreter) file is to be created by an editor and saved as a ".COM" file. (Other DEC operating systems have similar structures but the names and extensions may be different.) The procedure as written assumes that the executable files for the programs exist on the system device "SY:" and that the data resides on the default device "DK:". The output of the programs goes to the DEC line printer device. If the target system does not have a line printer, it will be

necessary to assign the line printer device to the terminal or to change the programs to "TYPE" the output instead of "PRINT". In any case, the prompts currently use the "PRINT" statement and this will have to be changed to TYPE if one wants to use the terminal and the line printer at the same time. Also, the "READ" statements for terminal input will have to be changed to "ACCEPT" statements unless they are redirected through appropriate ASSIGN commands in RT-11. There are also some minor differences between the UNIX and DEC forms of the Fortran OPEN statement. It may be necessary to make some modifications to these statements if RT-11 Fortran IV is to be used. If any of the DEC Fortran-77 compilers is used, however, there should not be any necessity for changing the OPEN statements which correspond to the ANSI standard for Fortran-77.

APPENDIX C: LINE-OF-SIGHT GEOMETRY FOR UVSP OCCULTATION EXPERIMENTS

The inversion of the UVSP occulation results requires precise determination of the pointing of the instrument during the occultation. Assuming, that the satellite position as a function of time is known to the required degree of accuracy, one may use the pitch, yaw and roll angles as received from the telemetry stream to derive the necessary pointing data. These angles are with respect to the instantaneous line-of-sight to the center of the solar disk and the solar axis.

It is most convenient to work in the familiar Cartesian geocentric corrdinate system defined by the x-axis pointing at the first Point of Aries and the positive z-axis through the North pole. We shall call this the GCS system. The solar coordinates are obtained from a 22 term Chebyshev fit as published by the Naval Observatory. The satellite ephemeris is provided on tape in the GCS system as well. The unit vector of the reference line of sight to the center of the solar disk is given by

$$\hat{e}_o = (\vec{R}_{sun} - \vec{r}_{sat})/|\vec{R}_{sun} - \vec{r}_{sat}| \qquad (1)$$

where $\vec{r}_{sun}$ and $\vec{r}_{sat}$ are the vectors defining respectively the solar and satellite positions in the GCS system.

Any point $\vec{x}$ along the line of sight $\hat{e}_o$ may be represented in parametric form the relation

$$\vec{r} = \vec{r}_{sat} + t\,\hat{e}_o \qquad (2)$$

where t is the parameter. It is a simple matter to determine the value of t which corresponds to the tangent point. Note that this t is the magnitude of the distance from the satellite to the tangent point. With this value of t one can easily compute the vector of the tangent point along the $\hat{e}_o$ line of sight, $\vec{x}_{tan}$ using Equation (2).

We now need to determine the pointing of the sensor more precisely. The line of sight is specified by three angles, pitch (ξ), yaw (η) and roll (ω) with respect to the projection of the solar pole and equator on the solar disk. This can be handled more simply in a coordinate system with origin at the satellite. To transform to the new system we define a set of rotation matrices. First a rotation through ϕ_{sat} followed by a rotation through ($\pi/2 - \lambda_{sat}$) where

$$\phi_{sat} = \tan^{-1}(y_{sat}/x_{sat}) \tag{3}$$

and

$$\lambda_{sat} = \tan^{-1}(y_{sat}/[x_{sat}^2 + y_{sat}^2]^{1/2}) \tag{4}$$

The rotation matrix is

$$R_1 = \begin{pmatrix} \sin\lambda_{sat} & 0 & -\cos\lambda_{sat} \\ 0 & 1 & 0 \\ \cos\lambda_{sat} & 0 & \sin\lambda_{sat} \end{pmatrix} \begin{pmatrix} \cos\phi_{sat} & \sin\phi_{sat} & 0 \\ -\sin\phi_{sat} & \cos\phi_{sat} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} \sin\lambda_{sat}\cos\phi_{sat} & \sin\lambda_{sat}\sin\phi_{sat} & -\cos\lambda_{sat} \\ -\sin\phi_{sat} & \cos\phi_{sat} & 0 \\ +\cos\lambda_{sat}\cos\phi_{sat} & +\cos\lambda_{sat}\sin\phi_{sat} & \sin\lambda_{sat} \end{pmatrix} \tag{5}$$

This is followed by a translation up the radius to the satellite by the distance

$$h_{sat} = x_{sat}^2 + y_{sat}^2 + z_{sat}^{2\ 1/2} \tag{6}$$

The next part of the transformation is an inversion of the transformed (x-y) plane which is equivalent to a rotation through π about the transformed z axis. The new y-axis is locally horizontal at the satellite. This is followed by a rotation about this latter y-axis through the zenith angle of the sun at the satellite. This angle is

$$\xi = \cos^{-1}(\hat{e}_o \cdot \vec{x}_{sat}/h_{sat}) \tag{7}$$

These two rotations are combined as

$$R_2 = \begin{pmatrix} \cos\xi & 0 & -\sin\xi \\ 0 & 1 & 0 \\ \sin\xi & 0 & \cos\xi \end{pmatrix} \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} -\cos\xi & 0 & -\sin\xi \\ 0 & -1 & 0 \\ -\sin\xi & 0 & \cos\xi \end{pmatrix} \tag{8}$$

We now have a coordinate system whose z-axis points toward the center of the disk and whose y axis is really horizontal along the line of sight and whose x axis is parallel to the positive vertical at the tangent height. We

shall erect a plane at the tangent point perpendicular to the line of sight and project the disk onto this plane. To define the axis on this projected plane we need to know the direction of the solar rotation axis on this plane.

This direction is defined by three quantities: the angle of the ecliptic plane with the equator $\varepsilon = 23^{o}\ 26'$, the tilt of the polar axis with respect to the ecliptic $\alpha = 7^{o}\ 15'$ and the longitude of the ascending node of the solar equator on the ecliptic plane. This latter quantity is defined with a secular term to account for the earth's nutation and other minor variations

$$\beta = 73^{o}\ 40' + 50.25''\ t \tag{9}$$

where t is time in years after 1850.

What we need is χ, the angle that the solar axis makes with the z-axis of the GCS. For any given value of t in Equation (9) this will be constant. Clearly for one occultation experiment, t is also effectively constant and thus we shall ignore any changes in this angle χ as well. It is a simple matter to derive the transformation from the GCS to the solar rotation system as the set of rotations

$$R_s = \begin{pmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{pmatrix} \begin{pmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\varepsilon & \sin\varepsilon \\ 0 & -\sin\varepsilon & \cos\varepsilon \end{pmatrix} \tag{10}$$

Using this transformation it is simple to show that the angle χ is defined by

$$\cos\chi = \cos\beta\cos\varepsilon + \sin\alpha\sin\beta\sin\varepsilon \tag{11}$$

and the unit vector in the GCS system in the direction of the solar rotation axis is

$$\hat{U}_{sun} = \begin{pmatrix} \sin\beta\cos\alpha \\ \sin\beta\sin\alpha\cos\varepsilon - \cos\beta\sin\varepsilon \\ \sin\beta\sin\alpha\sin\varepsilon + \cos\beta\cos\varepsilon \end{pmatrix} \tag{12}$$

Let $\hat{U}_s$ be the unit vector defining the solar rotation axis in the GCS. Then this vector may be transformed to the satellite system with z axis pointing at

the center of the solar disk with the axis locally horizontal at the satellite. The tangent point along this path is readily computed and one can easily define a great circle plane between the satellite and the tangent point of the solar center line of sight. We now erect a vertical plane at the tangent point which, by definition, is perpendicular to the line of sight (z-axis). The unit vector $\hat{U}_s$ may be transformed to this system by applying the transformations already applied to the line of sight unit vector ($\hat{e}_o$), namely R_1 and R_2. Thus

$$\hat{U}_s' = R_2 R_1 \hat{U}_s \equiv (\cos\gamma_1, \cos\gamma_2, \cos\gamma_3)^T \tag{13}$$

The angle between the projection of $\hat{U}'_s$ on the $(x' - y')$ plane and the x' axis is just

$$\tan\phi' = \cos\gamma_2/\cos\gamma_1 \tag{14}$$

This defines the orientation of the solar rotation axis on the projected plane. To get the desired orientation we now define another rotation matrix to bring the x' axis in line with the projected solar axis of rotation namely

$$R_{ax} = \begin{pmatrix} \cos\phi' & \sin\phi' & 0 \\ -\sin\phi' & \cos\phi' & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{15}$$

The actual pointing of the instrument is defined by three angles, pitch (ξ), yaw (η), and roll (l). The pitch is positive toward the south solar pole, the yaw is positive toward the solar east and the roll is clockwise from the projected solar north pole. For a given pitch (ξ), Yaw (η) combination the center of the slit is located at

$$(x'', y'') = D_t(-\sin\xi, -\tan\eta) \tag{16}$$

where D_t is the distance from the satellite to the tangent point of reference. For the narrow aeronomy slit (V, no. 20), we assume it is accurately represented by a line (1" x 180"). The end points of this line are then given by

$$(x''(\pm), y''\pm) = (x''\pm D_t \tan(90'')\sin\omega, y''\pm D_t \tan(90'')\cos\omega) \tag{17}$$

One can easily check whether either end point is beyond the defined edge of

the disk by adding into the solar ephemeris, the expression for the apparent solar diameter.

For simplicity of discussion let us calculate the center of the projected slit (x", y"). The other aspects are computed the same way as needed. The 3-vector to the center of the slit is $(x", y", (x"^2 + D_t^2)^{1/2})$. This then defines a unit vector $\hat{e}_o$" for the line of sight of the measurement. This can now be transformed to the GCS by the series of transformations

$$\hat{e} = R_1^T R_2^T R_{ax}^T \hat{e} \tag{18}$$

and this unit vector can be used to compute the tangent point for the true line of sight.

We now turn to the calculation of the latitude and longitude of the satellite and the tangent point. The observation is made at a given timme of day on a given dav of the year. The latitude of any point (x,y,z) is

$$\tan\lambda = z/(x^2 + y^2)^{1/2} \tag{19}$$

To compute the longitude of the point, one may use the relation

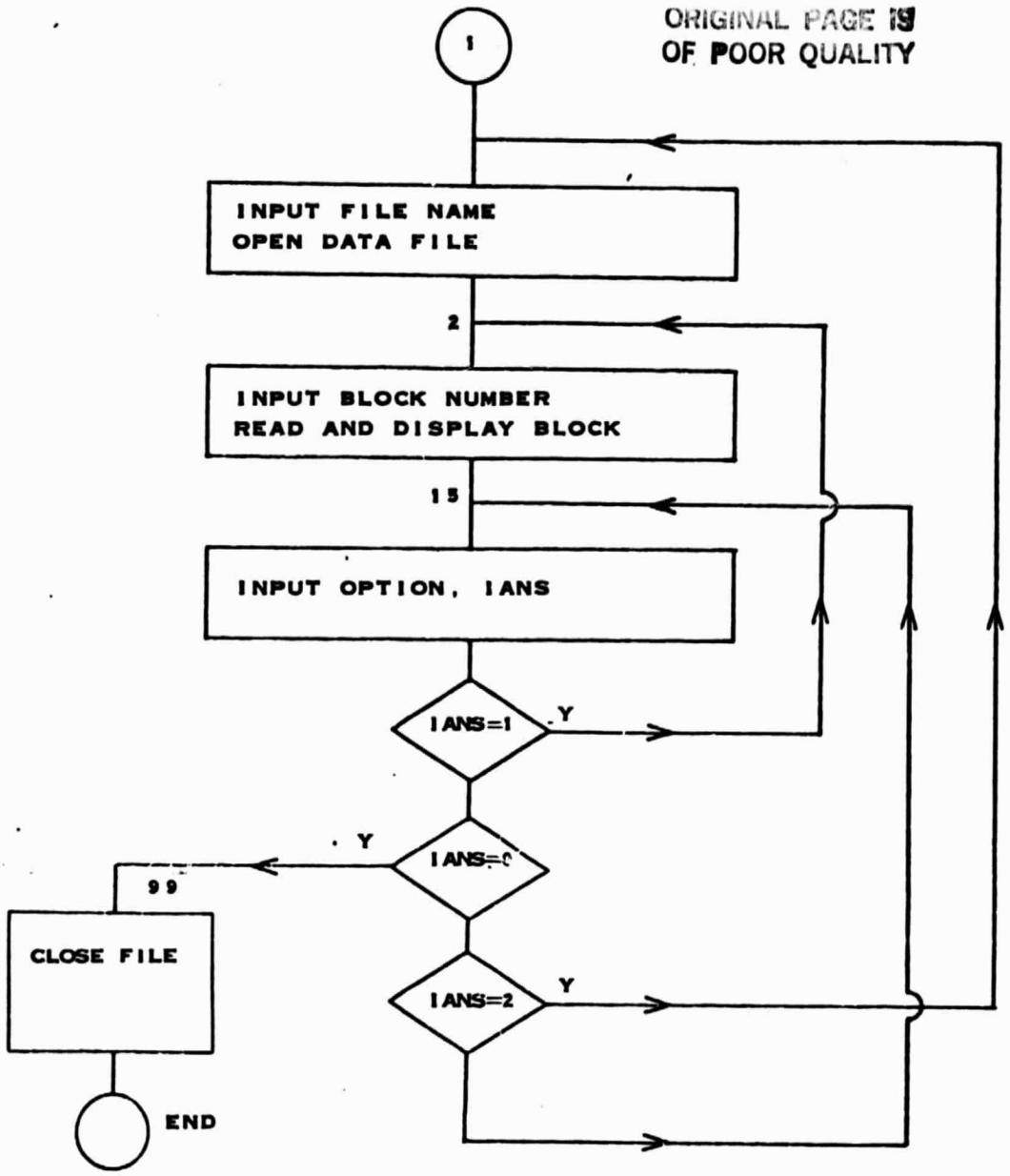
$$\cos\zeta_1 = \sin\delta \sin\lambda + \cos\sigma \cos\lambda \cosh \tag{20}$$

where ζ_1 is zenith angle of the sun, δ is the solar declination, λ is the latitude of the point and h is the hour angle with respect to the sun at the point. This can be solved for h to within a sign ambiguity. The sign is positive for sunset and negative for sunrise. The hour angle of the point is

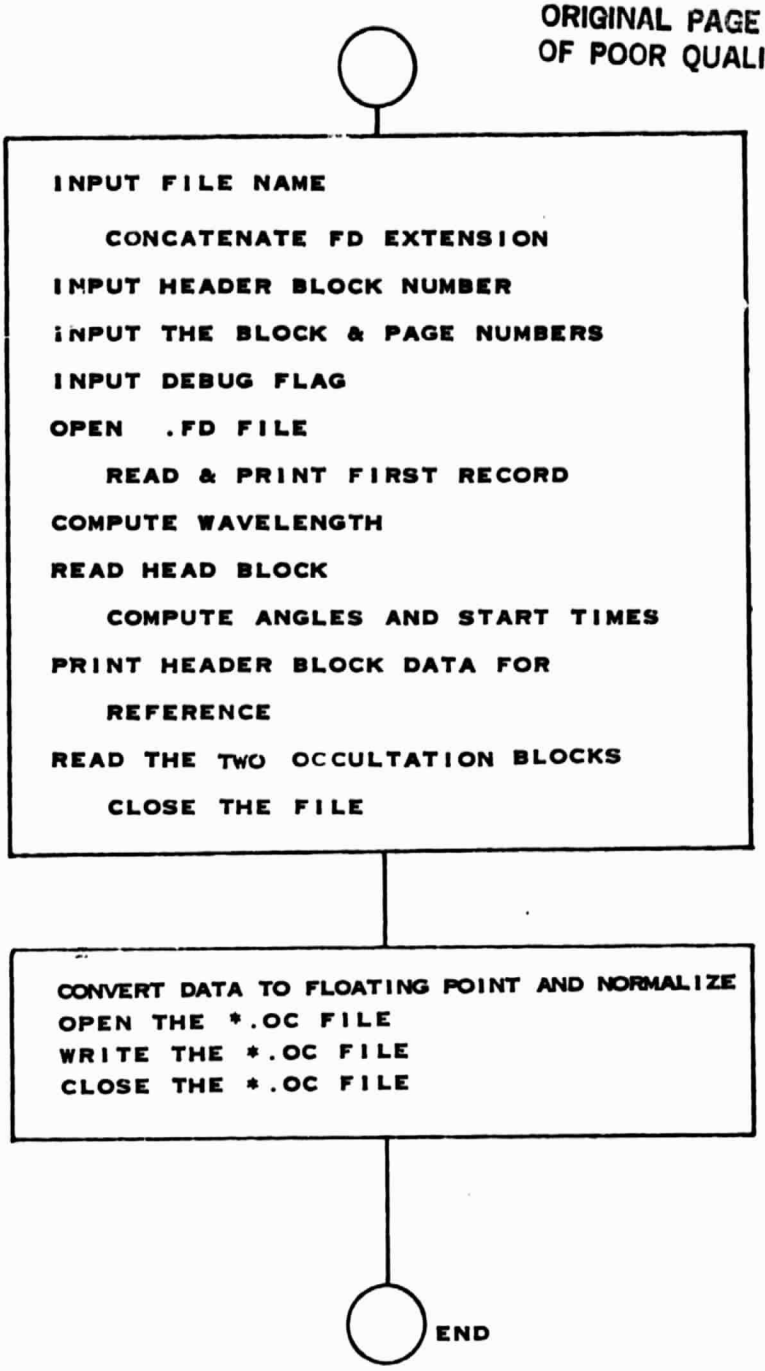
$$h = \cos^{-1} [(\cos\zeta_1 - \sin\delta \sin\lambda) / \cos\delta \cos\lambda] \tag{21}$$

The longitude is then

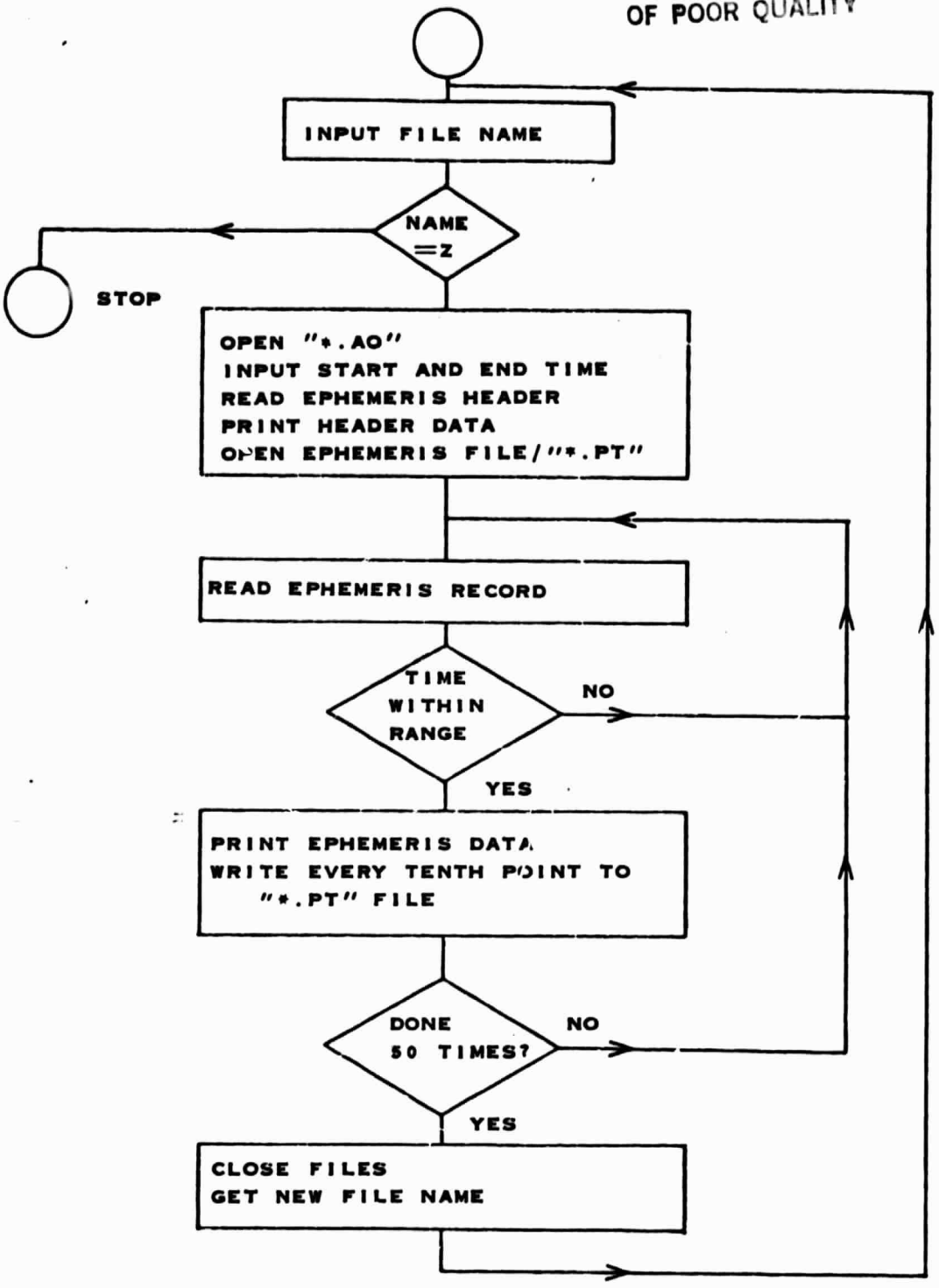
$$l = (UT/24)*2\pi - \pi + h \tag{22}$$



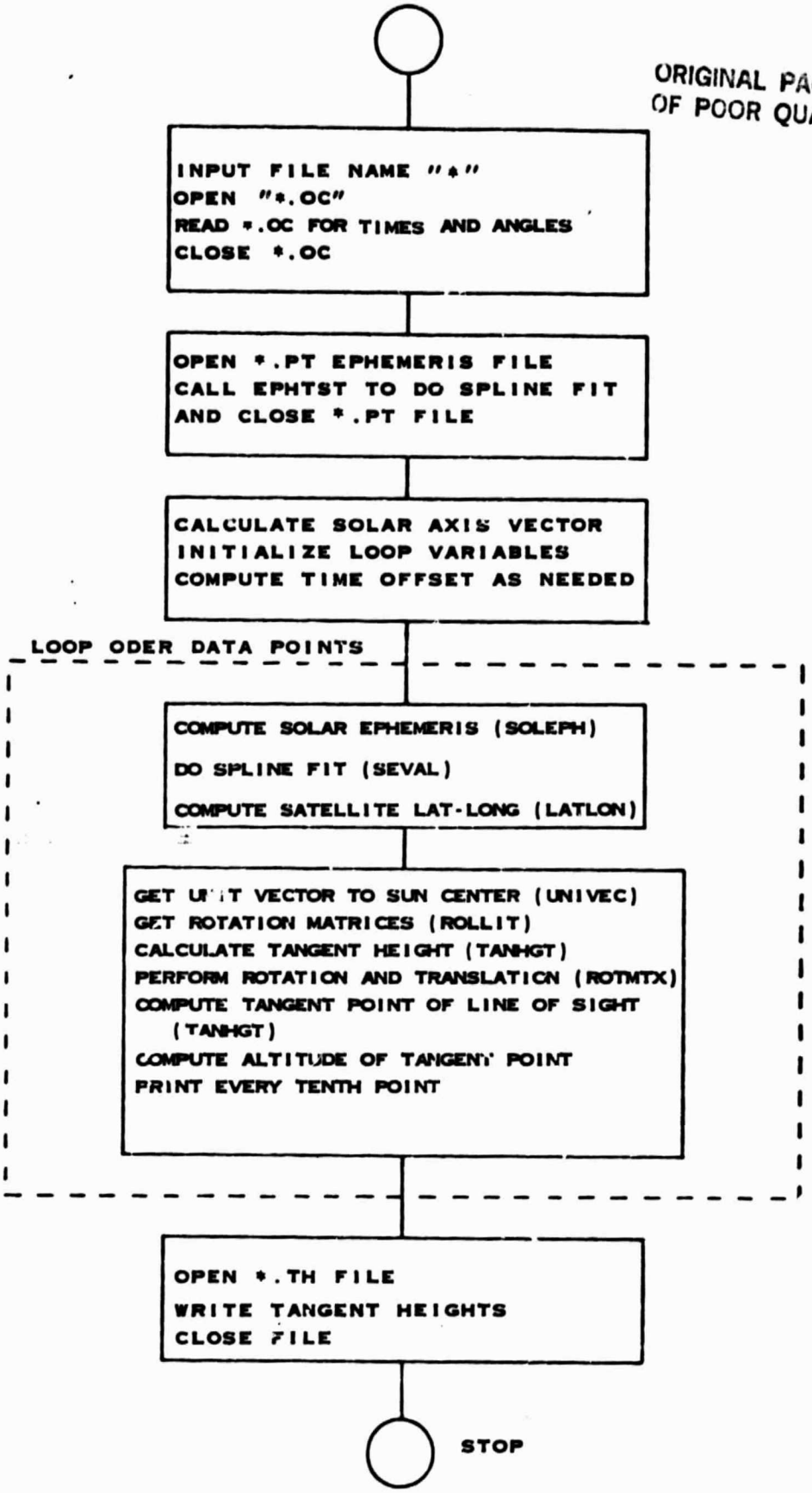
PROGRAM DATFND



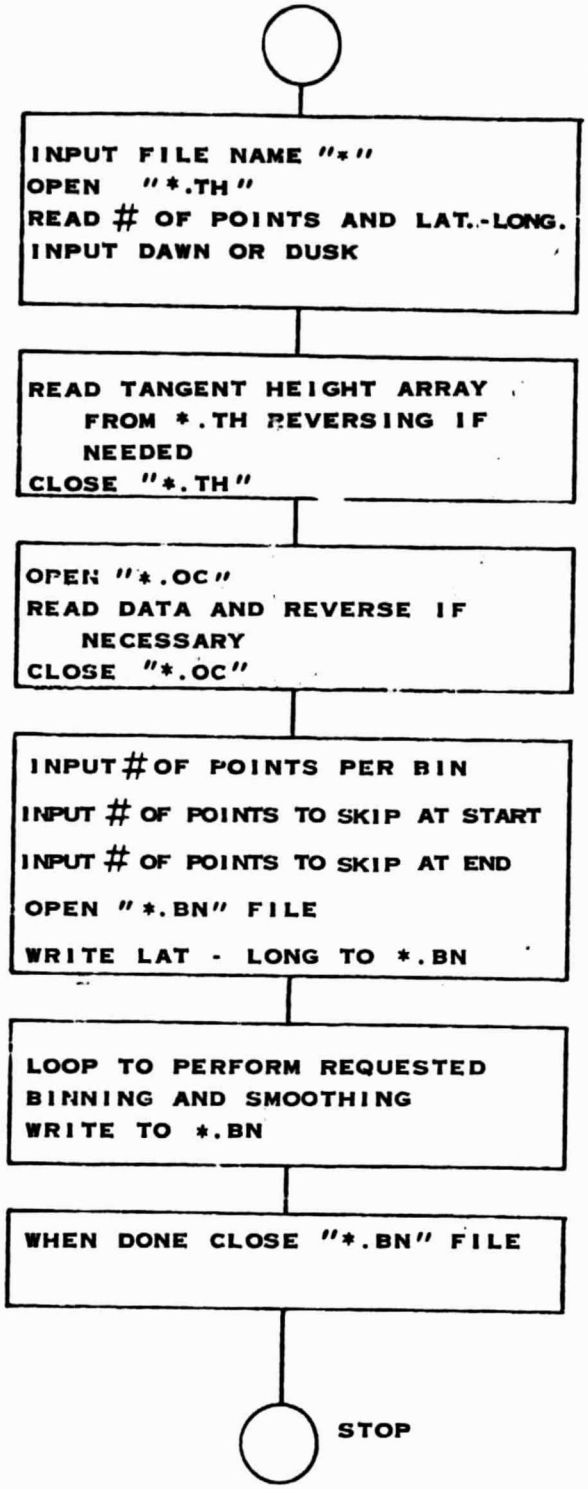
PROGRAM DATPGM



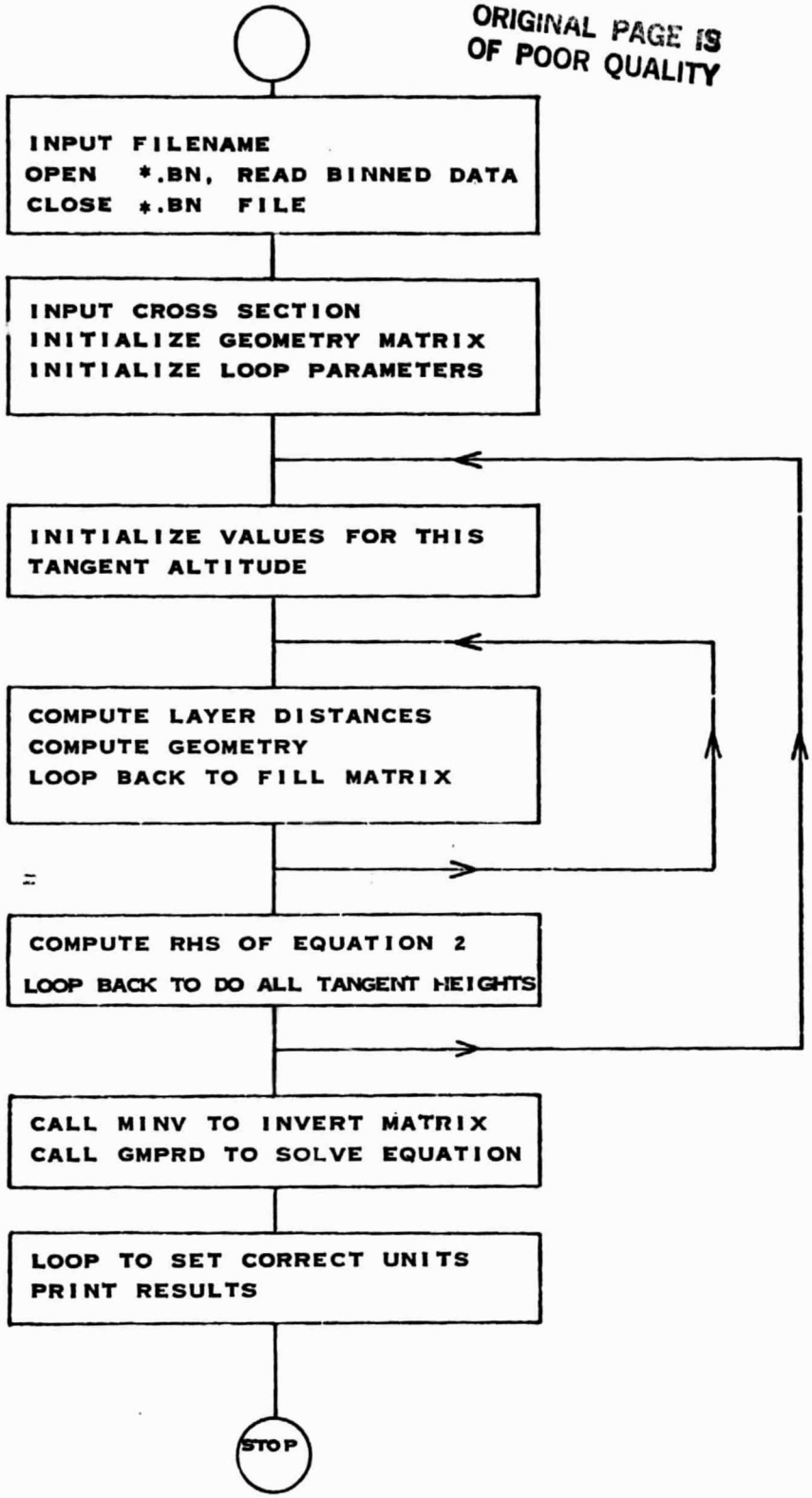
PROGRAM AOTOPT



PROGRAM TANSMM



PROGRAM BINS



PROGRAM REALOZ